PEARSON EDEXCEL INTERNATIONAL A-LEVEL

Decision Mathematics 1 (WDM11)

核心知识点与 2026 年 1 月最新真题精讲

学习笔记、算法归纳及易错考点总结

编撰：Tim Hu

2026 年 5 月 11 日

"Algorithms are the computational poetry of the logical mind."

Empowering your path to A* in A-Level Mathematics

目录

第 1 章 Algorithms (算法基础) **1**

1.1 Binary Search (二分查找) 1

1.2 Sorting Algorithms (排序算法) 2

1.3 Bin-Packing Algorithms (装箱算法) 5

第 2 章 Graphs and MST (图与最小生成树) **9**

2.1 Graph Theory Fundamentals (图论基础概念) 9

2.2 Minimum Spanning Tree Algorithms (最小生成树算法) 10

第 3 章 Algorithms on Graphs 图论算法 **15**

3.1 Dijkstra's Algorithm (最短路径) 15

3.2 Route Inspection Algorithm (路线巡检 / 中国邮递员问题) 17

3.3 The Travelling Salesman Problem (TSP 旅行商问题) 22

第 4 章 Critical Path Analysis (关键路径分析) **26**

第 5 章 Linear Programming (线性规划) **32**

A Appendix 1: Glossary (核心专业词汇总结) **36**

B Appendix 2: Algorithm Index (算法速查索引) **38**

广州市菲特思学校
Fettes College
Guangzhou

第 1 章 Algorithms (算法基础)

算法 (Algorithm) 是为了解决某个特定问题而执行的有限步骤序列。本章核心考察三种场景：在有序列表中搜索、对无序列表排序、以及将物品最优地装入固定容量的箱子中。

1.1 Binary Search (二分查找)

Key Point 核心知识点 (Binary Search Preconditions)

二分查找的前提：列表必须首先是 **有序的 (ordered)**。如果题目给出的列表是无序的，必须先将其排序或明确指出无法直接使用二分查找。

Algorithm 算法模型 (Binary Search 二分查找算法)

二分查找通过不断将搜索范围减半来定位目标项 (Target item)。

1. 假设当前搜索列表有 n 项。使用公式 $\frac{n+1}{2}$ 来寻找中点位置。如果结果带有小数，则 **向上取整 (round up)**。找到位于该位置的项作为 **Pivot**。
2. 如果 Pivot 正是你要找的 Target，查找成功，停止。
3. 如果不是，则根据大小关系判断：**舍弃 Pivot 本身，以及 Target 不可能存在的那一半列表**。
4. 在剩下的子列表中重复步骤 1 3。如果子列表为空仍未找到，则得出结论：Target 不在列表中。

Example 1.1 (Paper 1, Jan 2026, Q1 | Binary Search)

In the following list, eleven sports are identified by their initial letter:

(V) Volleyball (H) Handball (W) Wheelchair basketball (B) Baseball (R) Rugby (G) Golf (C) Cricket (P) Paratriathlon (T) Tennis (D) Diving (S) Squash

(a) Explain why a binary search cannot be performed on this list in its current form. (1 Mark)

(b) Write the list in an appropriate form and show how a binary search is used to try to locate (F) Football. You must make your choice of pivots clear. (4 Marks)

Solution.

Part (a):

The list is not in alphabetical order. (二分查找要求列表必须首先是有序的)。

Part (b):

先将字母按字母表升序 (Alphabetical order) 排列。在每一轮查找中，我们利用公式 $\frac{n+1}{2}$（遇小数向上取整）来确定 Pivot（基准点），并用**粗体**标出：

第 1 次查找： 剩余列表：B C D G H **P** R S T V W

共有 11 项，中点位置为 $\frac{11+1}{2} = 6$，故 Pivot 是第 6 项 **P**。

因为 F 在 P 之前，所以舍弃 P 及其右侧所有的项。

第 2 次查找： 剩余列表：B C **D** G H

共有 5 项，中点位置为 $\frac{5+1}{2} = 3$，故 Pivot 是第 3 项 **D**。

因为 F 在 D 之后，所以舍弃 D 及其左侧所有的项。

第 3 次查找： 剩余列表：G **H**

共有 2 项，中点位置为 $\frac{2+1}{2} = 1.5 \implies 2$ (向上取整)，故 Pivot 是第 2 项 **H**。

因为 F 在 H 之前，所以舍弃 H 及其右侧所有的项。

第 4 次查找： 剩余列表：**G**

共有 1 项，中点位置为 $\frac{1+1}{2} = 1$，故 Pivot 是 **G**。

因为 F 在 G 之前，所以舍弃 G。

结论：

列表已为空。**Football is not in the list.**

1.2 Sorting Algorithms (排序算法)

Key Point 核心知识点 (Comparison of Sorting Algorithms)

- **Bubble Sort (冒泡排序)**: 易于理解和实现。可以自动检测列表是否已经排好序（如果一遍 Pass 下来没有任何交换即可提前停止）。但在处理长列表时效率极低。
- **Quick Sort (快速排序)**: 使用分治法 (Divide and Conquer)，对于长列表效率通常高得多。但手动推导时稍显复杂，容易因为未保持剩余元素的“原序”而失分。

Algorithm 算法模型 (Bubble Sort 冒泡排序)

1. 从左到右，依次比较列表中的 **相邻两项 (adjacent items)**。
2. 如果它们顺序正确，则保留原样 (leave them)；如果顺序颠倒，则 **交换它们 (swap them)**。
3. 完成一整轮比较称为一个 **Pass**。此时列表中最大（或最小）的项必然“冒泡”到了最后

一位。
4. 重复上述步骤。当一个完整的 Pass 中 **没有任何交换 (without any swaps)** 发生时，排序完成。

Example 1.2 (Paper 1, Jan 2026, Q4 (a)(b) | Bubble Sort)

The following list of eleven distinct integers is to be sorted into ascending order:

$$44 \quad 35 \quad 73 \quad x \quad 24 \quad 58 \quad 32 \quad 21 \quad 61 \quad 54 \quad 31$$

The first two passes of bubble sort are:

1st: 35 44 x 24 58 32 21 61 54 31 73
2nd: 35 44 24 x 32 21 58 54 31 61 73

(a) Explain why x must satisfy $45 \leq x \leq 57$. (2 Marks)
(b) Carry out one more pass of bubble sort, showing the result at the end of the pass. (2 Marks)

Solution.

Part (a):
观察第二遍 (2nd pass) 的结果，44 与 x 在扫描时没有交换位置（依然是 44 在 x 前面），这说明 $x > 44$。
随后扫描到 x 与 58 时，也没有交换位置（x 依然在 58 前面），这说明 $x < 58$。
因为题目指明所有数字是 **distinct integers (互不相同的整数)**，所以 x 只能取 45 到 57 之间的整数，即 $45 \leq x \leq 57$。
Part (b):
基于 2nd pass 的结果，继续从左到右两两比较：
35 和 44 (不换) → 44 和 24 (换) → 44 和 x (因 $x \geq 45$，不换) → x 和 32 (换) → x 和 21 (换) → x 和 58 (不换) → 58 和 54 (换) → 58 和 31 (换) → 58 和 61 (不换) → 61 和 73 (不换)。
End of 3rd pass:

$$35 \quad 24 \quad 44 \quad 32 \quad 21 \quad x \quad 54 \quad 31 \quad 58 \quad 61 \quad 73$$

Algorithm 算法模型 (Quick Sort 快速排序)

1. 选择列表的中点项作为第一个 **Pivot**。使用公式 $\frac{n+1}{2}$ 并 **向上取整 (round up)**。例如，若有 10 项，则中点为 5.5，取第 6 项。
2. 将所有 **小于** Pivot 的项写在一个子列表中，**切记必须保持它们在原列表中的先后顺序 (Do not sort the items as you write them down)**。
3. 写下 Pivot。
4. 将所有 **大于** Pivot 的项写在另一个子列表中，同样保持原序。
5. 对生成的每个子列表重复步骤 1-4，直到所有元素都曾被选为 Pivot。

注：*Edexcel 考试中，必须清晰标出每一行的 Pivot，通常通过加下划线或加粗表示。*

Example 1.3 (Paper 1A, Jan 2026, Q1(c) | Quick Sort)

The numbers in the list are the lengths, in feet, of eleven pieces of wood:

5.0 1.8 6.2 2.8 3.0 4.0 3.8 2.4 0.6 0.8 7.8

Carry out a quick sort to produce a list of the lengths in **descending (降序)** order. You should show the result of each pass and identify your pivots clearly. (4 Marks)

Solution.

因为是降序，所以分为子列表时，**大于** Pivot 的放左边，**小于** Pivot 的放右边。
11 项的中点为 $\frac{11+1}{2} = 6$，所以首个 Pivot 为第 6 项 **4.0**。

5.0	1.8	6.2	2.8	3.0	**4.0**	3.8	2.4	0.6	0.8	7.8	Pivots
5.0	**6.2**	7.8	4.0	1.8	2.8	3.0	**3.8**	2.4	0.6	0.8	6.2, 3.8
7.8	6.2	5.0	4.0	3.8	1.8	2.8	3.0	**2.4**	0.6	0.8	(7.8, 5.0), 2.4
7.8	6.2	5.0	4.0	3.8	2.8	**3.0**	2.4	1.8	**0.6**	0.8	3.0, 0.6
7.8	6.2	5.0	4.0	3.8	3.0	2.8	2.4	1.8	**0.8**	0.6	(2.8), 0.8
7.8	6.2	5.0	4.0	3.8	3.0	2.8	2.4	1.8	0.8	0.6	

(注：粗体表示本轮选中的 *Pivot*，下划线表示已经被作为 *Pivot* 确立最终位置的项。在生成子列表时，务必保持元素原有相对顺序。)

最终顺序: 7.8 6.2 5.0 4.0 3.8 3.0 2.8 2.4 1.8 0.8 0.6

1.3 Bin-Packing Algorithms (装箱算法)

Key Point 核心知识点 (Comparison of Bin-packing Algorithms)

考纲要求掌握以下三种装箱算法的优缺点分析:

Type of Algorithm	Advantage (优点)	Disadvantage (缺点)
First-fit	Quick to apply (应用快速)	Not likely to lead to a good solution (通常得不到最优解)
First-fit decreasing (FFD)	Usually a good solution; easy to apply (通常能得到良好解，且易于操作)	May not get an optimal solution (不一定能得到最优解)
Full-bin	Usually a good solution (解通常极佳)	Difficult to do, especially when the numbers are plentiful or awkward (数据量大或凑整困难时很难操作)

Lower Bound (下界) 的计算: Lower bound $= \frac{\text{Sum of all items}}{\text{Bin size}}$，计算结果必须 **向上取整 (round up)**。

Algorithm 算法模型 (Bin-Packing Algorithms 装箱算法)

1. First-fit algorithm (首次适应算法):

- 按照题目给定的原始顺序 (Take the items in the order given) 依次处理每个物品。
- 每次都从 **Bin 1** 开始检查，将物品放入第一个能容纳它的箱子中。

2. First-fit decreasing algorithm (FFD 降序首次适应算法):

- 先将所有物品按 **从大到小 (descending order)** 排序。
- 然后对重新排序后的列表应用 First-fit 算法。

3. Full-bin packing (满箱算法):

- 通过观察 (Use observation) 找出能正好填满一个箱子 (fill a bin completely) 的物品组合，优先将这些组合装箱。
- 将剩余的物品使用 First-fit 算法装入后续的箱子中。

Example 1.4 (Paper 1A, Jan 2026, Q1 | First-Fit & FFD Packing)

The numbers in the list are the lengths, in feet, of eleven pieces of wood. They are to be cut from planks of wood of length 10 feet. You should ignore wastage due to cutting.

5.0 1.8 6.2 2.8 3.0 4.0 3.8 2.4 0.6 0.8 7.8

(a) Calculate a lower bound for the number of planks needed. You must make your method clear. (2 Marks)
(b) Use the first-fit bin packing algorithm to determine how these pieces could be cut from 10 feet planks. (3 Marks)
(d) Use the first-fit decreasing bin packing algorithm to determine how these pieces could be cut from 10 feet planks. (2 Marks)

Solution.

Part (a): Lower Bound Calculation

Total Sum = $5.0 + 1.8 + 6.2 + 2.8 + 3.0 + 4.0 + 3.8 + 2.4 + 0.6 + 0.8 + 7.8 = 38.2$

Lower bound = $\frac{38.2}{10} = 3.82 \implies$ 向上取整，最少需要 **4 planks**。

Part (b): First-Fit Algorithm

按原始顺序直接装箱，每次从 Bin 1 开始尝试：

- 5.0 → Bin 1
- 1.8 → Bin 1 (目前 Bin 1: 6.8)
- 6.2 → Bin 1 装不下，放入 Bin 2
- 2.8 → Bin 1 装得下 ($6.8 + 2.8 = 9.6 \le 10$)，放入 Bin 1
- 3.0 → Bin 1 放不下，Bin 2 可以 ($6.2 + 3.0 = 9.2 \le 10$)，放入 Bin 2
- 4.0 → Bin 1, 2 放不下，放入 Bin 3
- 3.8 → 放入 Bin 3 (目前 Bin 3: 7.8)
- 2.4 → Bin 1, 2, 3 均放不下，放入 Bin 4
- 0.6 → Bin 1 放不下，Bin 2 可以 ($9.2 + 0.6 = 9.8 \le 10$)，放入 Bin 2
- 0.8 → 放入 Bin 3 ($7.8 + 0.8 = 8.6 \le 10$)
- 7.8 → Bin 1 4 均放不下，放入 Bin 5

最终结果 (First-Fit)：

Bin 1: 5.0, 1.8, 2.8

广州市斐特思学校 Fettes College Guangzhou

Bin 2: 6.2, 3.0, 0.6
Bin 3: 4.0, 3.8, 0.8
Bin 4: 2.4
Bin 5: 7.8

Part (d): First-Fit Decreasing (FFD) Algorithm

首先对列表降序排列 (利用前序小题的结果)：

7.8 6.2 5.0 4.0 3.8 3.0 2.8 2.4 1.8 0.8 0.6

然后执行 First-Fit：

- **Bin 1:** 7.8, 1.8 (总和 9.6，剩余 0.4)
- **Bin 2:** 6.2, 3.8 (总和 10.0，剩余 0)
- **Bin 3:** 5.0, 4.0, 0.8 (总和 9.8，剩余 0.2)
- **Bin 4:** 3.0, 2.8, 2.4, 0.6 (总和 8.8，剩余 1.2)

最终结果 (FFD)：恰好使用 4 个 Bin，达到了下界的最优解。

Example 1.5 (Paper 1, Jan 2026, Q4 | Reverse-engineering Packing)

Given that when the numbers[35, 24, 44, 32, 21, x, 54, 31, 58, 61, 73] are placed in bins of size 120, the lower bound for the number of bins required is five.
When the first-fit decreasing algorithm is used to place the numbers in bins of size 160, x is placed in the second bin, which also contains 54.
Determine the value of x. You must make your reasoning clear. (3 Marks)

Solution.

步骤 1：利用 Lower Bound 求出 x 的范围

已知 Lower bound = 5，箱子容量 = 120，故所有数字总和必大于 $4 \times 120 = 480$。

已知数字之和 $= 35 + 24 + 44 + 32 + 21 + 54 + 31 + 58 + 61 + 73 + x = 433 + x$。

故 $433 + x > 480 \implies x > 47$。

结合该题前序小问中得出的结论 ($45 \le x \le 57$)，此时可缩小范围为 $\mathbf{48 \le x \le 57}$。

步骤 2：利用 FFD 的装箱过程锁定 x 的精确值

在 FFD (Bin size 160) 中，需先将数字降序排列。因为 $x \ge 48$，所以前几个确定的数字为：73, 61, 58, 54, x (由于 54 之后可能有其他数字，但 x 一定在 54 之后或其附近)...

依次装入箱子 (箱子容量为 160)：

- **Bin 1:** 先放 73，再放 61。此时 $73 + 61 = 134$，剩余容量 $160 - 134 = 26$。下一个最大的物品 58 放不下。往后找，24 可以放进去 ($134 + 24 = 158 \leq 160$)。所以 **Bin 1: 73, 61, 24**。
- **Bin 2:** 接下来剩下的最大数字是 58。放入 58，再放 54。此时 $58 + 54 = 112$。
  题目明确指出 "x is placed in the second bin, which also contains 54"。
  这意味着装完 54 后，紧接着（或稍后）把 x 也装进了 Bin 2。
  那么必须满足容量限制：$58 + 54 + x \leq 160 \implies 112 + x \leq 160 \implies \mathbf{x \leq 48}$。

综合步骤 1 ($x \geq 48$) 和步骤 2 ($x \leq 48$)，必定有 $\mathbf{x = 48}$。

广州市斐特思学校
Fettes College
Guangzhou

第 2 章 Graphs and MST (图与最小生成树)

2.1 Graph Theory Fundamentals (图论基础概念)

Key Point 核心知识点 (Graph Basics & Traversal (图的基础与遍历))

1. 基础元素 (Basic Elements):

- **Graph (图)**: 由 **vertices / nodes (顶点)** 和连接它们的 **edges / arcs (边/弧)** 组成。
- **Weighted graph / Network (加权图/网络)**: 每条边都有相关联的数字（通常称为 weight 权重）。
- **Subgraph (子图)**: 原图的一部分，其顶点和边都属于原图。
- **Degree / Valency / Order (度)**: 连接到某个顶点的边的数量。度数为偶数称为 **even degree**，奇数称为 **odd degree**。
- **Euler's Handshaking Lemma (欧拉握手引理)**: 无向图中，所有顶点的度数之和等于边数的 2 倍。(推论：任何图中奇数度顶点的个数必定是偶数)。

2. 路径与环 (Walks, Paths and Cycles):

- **Walk**: 沿着边从一个顶点到另一个顶点的路线。
- **Path (路径)**: 任何 **顶点 (vertex)** 都不被访问超过一次的 walk。
- **Trail (轨迹)**: 任何 **边 (edge)** 都不被访问超过一次的 walk。
- **Cycle (环/回路)**: 起点和终点相同，且除此之外不重复访问任何其他顶点的 walk。
- **Hamiltonian cycle (哈密顿回路)**: 包含图中 **每一个顶点**的 cycle。

Key Point 核心知识点 (Trees, Special Graphs & Matrices (树与矩阵表示))

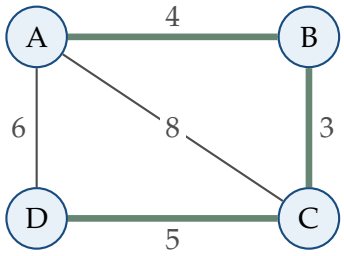
3. 树与特殊图 (Trees and Special Graphs):

- **Tree (树)**: 一个连通且 **没有环 (no cycles)** 的图。
- **Spanning tree (生成树)**: 包含原图所有顶点，且本身是一棵树的子图。
- **Complete graph (完全图)**: 每一个顶点都与其他所有顶点直接相连的图。
- **Isomorphic graphs (同构图)**: 展现相同信息（顶点连接关系完全一致）但画法不同的图。

4. 矩阵表示 (Matrix Representation):

- **Adjacency matrix (邻接矩阵)**: 矩阵中的数字表示连接对应顶点的 **边的数量 (number of arcs)**。

- **Distance matrix (距离矩阵)**: 矩阵中的数字表示对应边的 **权重 (weight)**，而不是边的数量。



- **Degree of A**: 3 (Odd degree)
- **Cycle**: e.g., A → B → C → A
- **Spanning Tree**: 绿色加粗部分 (连通所有节点且无环)

2.2 Minimum Spanning Tree Algorithms (最小生成树算法)

Algorithm 算法模型 (Kruskal's Algorithm 克鲁斯卡尔算法)

核心思想：全局贪心。从全图中最短的边开始挑选，只要不形成环就加入。

1. **Sort** all the arcs (edges) into ascending order of weight (将所有边按权重升序排列)。
2. **Select** the arc of least weight to start the tree (选择权重最小的边作为树的起点)。
3. **Consider** the next arc of least weight:
   - If it would form a **cycle** with the arcs already selected, **reject** it. (若形成环，则拒绝)
   - If it does **not** form a cycle, **add** it to the tree. (若不形成环，则加入树中)
   - *If there is a choice of equal arcs, consider each in turn.*
4. **Repeat** step 3 until all vertices are connected. (重复直至所有顶点被连通)

Algorithm 算法模型 (Prim's Algorithm on a Graph 图上的普里姆算法)

核心思想：局部贪心。以任意点为中心，不断向外扩张，只挑选连接“已在树中”和“未在树中”的最小边，无需事先排序。

1. **Choose** any vertex to start the tree (选择任意顶点作为起点)。
2. **Select** an arc of least weight that joins a vertex **already in the tree** to a vertex **not yet in the tree**.
   - *If there is a choice of arcs of equal weight, choose any of them.* (选择连接树内与树外顶点的最小边)
3. **Repeat** step 2 until all the vertices are connected. (重复直至所有顶点均被连通)

4. **List** the arcs in the order that they were added. (按加入顺序列出所有的边)

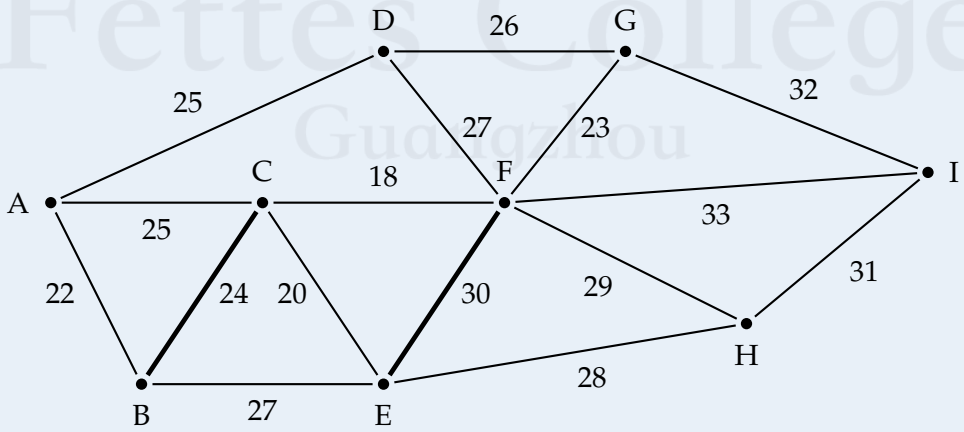
Algorithm 算法模型 (Prim's Algorithm on a Distance Matrix 距离矩阵上的普里姆算法)

考试常考在表格 (Matrix) 上直接执行 Prim 算法，需严格遵守以下打钩/画圈规则：

1. **Choose** any vertex to start the tree (选择任意起点)。
2. **Delete** the **row** in the matrix for the chosen vertex (划掉选中顶点对应的**行**)。
3. **Number** the **column** in the matrix for the chosen vertex (在选中顶点对应的**列**顶端标上数字序号，如 1, 2, 3...)。
4. **Circle** the lowest undeleted entry in the numbered columns. *If there is an equal choice, choose randomly.* (在所有已标序号的列中，圈出未被划掉的最小值)。
5. The circled entry becomes the next arc to be added to the tree (该圈出的值即为加入树的新边)。
6. **Repeat** steps 2, 3, 4 and 5 until all rows have been deleted.
7. **List** the arcs in the order that they were added.

Example 2.1 (Paper 1A, Jan 2026, Q2 | Prim & Kruskal Applications)

The figure represents nine buildings, A to I, and the cost of connecting alarm systems between them.



(a) Use **Prim's algorithm**, starting at A, to find the minimum spanning tree. You must list the arcs in the order that you select them. (3 Marks)

(c) It is discovered that alarm systems are already connected along **BC** and **EF**. Since these

already exist, it is decided to use these arcs as part of the spanning tree. Use **Kruskal's algorithm** to find the minimum spanning tree that includes arcs BC and EF. You must list the arcs in the order considered and state whether you are adding the arc to your spanning tree. (3 Marks)

Solution.

(a) Prim's Algorithm from A:
依据 Prim 的逻辑，每次从当前已在树内的节点中寻找通向外围节点的最短边。
按顺序选择：**AB, BC, CF, CE, FG, AD, EH, HI**。
(注：只要满足同权重的合法替换即可得分，例如到达 F 后向外延伸，最后找 H 和 I 的逻辑)
(c) Kruskal's with pre-existing arcs:
已经强制包含了 **BC, EF** (可视为权重最低，最先加入树中)。接下来的边按权重从小到大逐一审核：
- **CF** (18) → Add (此时树中含 BC, EF, CF)
- **CE** (20) → Reject (因为 C-F-E 结合已包含的 EF 会形成环)
- **AB** (22) → Add
- **FG** (23) → Add
- **AD** (25) → Add *(注：此处有两条 25 的边，先考虑哪条均可)*
- **AC** (25) → Reject (形成 A-B-C 环)
- **DG** (26) → Reject (形成 D-A-B-C-F-G 环)
- **BE** (27) → Reject (形成 B-C-F-E 环)
- **DF** (27) → Reject (形成 D-A-B-C-F 环)
- **EH** (28) → Add
- **FH** (29) → Reject (形成 F-E-H 环)
- **HI** (31) → Add

(注：总共 9 个顶点，加上初始的 2 条粗边，当挑选满 8 条边时算法结束。Kruskal 的考点在于必须写出被 Reject 的那些尝试项。)

广州市斐特思学校 Fettes College Guangzhou

Example 2.2 (Paper 1, Jan 2025, Q6 | Prim's Algorithm on Distance Matrix)

The table shows the shortest distances, in miles, between ten towns, A, B, C, D, E, F, G, H, J and K.

	A	B	C	D	E	F	G	H	J	K
A	-	16	26	19	22	34	30	41	45	36
B	16	-	10	15	38	33	40	25	29	20
C	26	10	-	12	39	30	31	15	19	10
D	19	15	12	-	33	18	25	28	31	22
E	22	38	39	33	-	15	8	33	20	29
F	34	33	30	18	15	-	7	24	19	28
G	30	40	31	25	8	7	-	25	12	21
H	41	25	15	28	33	24	25	-	13	5
J	45	29	19	31	20	19	12	13	-	9
K	36	20	10	22	29	28	21	5	9	-

(a) Explain the difference between the **classical** Travelling Salesman Problem and the **practical** Travelling Salesman Problem. (2 Marks)

(b) Use **Prim's algorithm**, starting at A, to obtain a minimum spanning tree for the network. You must clearly state the order in which you select the arcs of your tree. (3 Marks)

Solution.

Part (a): Theory

Classical problem must visit **every vertex exactly once** and returns to start, but **practical problem** must visit **every vertex at least once** and returns to start. (经典 TSP 必须“恰好”访问每个顶点一次；实际 TSP“至少”访问每个顶点一次，允许走捷径重复经过顶点)。

Part (b): Prim's Algorithm on Matrix

严格遵循“划行标列”的法则，寻找带有编号的列中未被划去的最小值：

- **Start at A**: 划掉行 A，在列 A 顶端标 1。列 A 中的最小值是 16。加入边 **AB**。
- **Node B added**: 划掉行 B，在列 B 顶端标 2。查看列 A, B 中未划掉的值，最小值是列 B 中的 10。加入边 **BC**。
- **Node C added**: 划掉行 C，列 C 标 3。查看列 A, B, C，最小值为列 C 中的 10。加入边 **CK**。
- **Node K added**: 划掉行 K，列 K 标 4。查看列 A, B, C, K，最小值为列 K 中的 5。加入边 **HK**。

- **Node H added**: 划掉行 H，列 H 标 5。查看前 5 列，最小值为列 K 中的 9。加入边 **JK**。
- **Node J added**: 划掉行 J，列 J 标 6。查看前 6 列，发现**存在两个并列的最小值 12**：分别位于列 C (对应边 **CD**) 和列 J (对应边 **GJ**)。这导致了两种合法的解题分支：

Branch 1 (先选 CD):

- 加入 **CD** (12)，划掉行 D，列 D 标 7。
- 此时列 J 中仍有 12，为当前最小。加入 **GJ** (12)，划掉行 G，列 G 标 8。
- 此时列 G 中暴露出更小的值 7。加入 **FG** (7)，划掉行 F，列 F 标 9。
- 最后，列 G 中有 8。加入 **EG** (8)，划掉行 E。所有行均被划掉，结束。
- **顺序: AB, BC, CK, HK, JK, CD, GJ, FG, EG**

Branch 2 (先选 GJ):

- 加入 **GJ** (12)，划掉行 G，列 G 标 7。
- 注意！一旦划掉行 G，列 G 中立刻暴露出了更小的值 7 (对应 **FG**)，优先选取。加入 **FG** (7)，划掉行 F，列 F 标 8。
- 接着，列 G 中还有 8。加入 **EG** (8)，划掉行 E，列 E 标 9。
- 最后剩下未选的最小值是之前列 C 里的 12。加入 **CD** (12)，划掉行 D。结束。
- **顺序: AB, BC, CK, HK, JK, GJ, FG, EG, CD**

(注：在考试中，写出以上任意一条边序列均可获得满分 3 分。必须严格按加入的顺序列出边的名称。)

广州市菲特思学校 Fettes College Guangzhou

第 3 章 Algorithms on Graphs 图论算法

3.1 Dijkstra's Algorithm (最短路径)

Key Point 核心知识点 (Dijkstra Basics)

- **作用**: 用于寻找网络中两点之间（或起点到沿途任意节点）的最短路径。也适用于带有单行道 **(Directed arcs)** 的网络。
- **节点框 (Box representation)**: 每个节点被替换为一个分为三块的矩形框：

Vertex	Order of labelling	Final label
Working values		

- **Working value (临时值)**: 随时更新，当发现更短的路径时覆盖原值。
- **Final label (最终值)**: 从所有 Working values 中挑选出的**最小值**，一旦确定即不可更改，代表起点到该点的最短距离。

Algorithm 算法模型 (Dijkstra's Algorithm)

1. 将起点 (Start vertex) 的 Final label 设为 0，Order 设为 1。
2. 找到刚获得 Final label 的点 X。对于所有与 X 直接相连的未确定点 Y，计算：

$$\text{New Working Value} = \text{Final label at } X + \text{weight of arc } XY$$

将其记录在 Y 的下方。

3. 在全图**所有**未确定 Final label 的点中，选出 Working value **最小**的一个，将其升级为 Final label，并标上 Order。
4. 重复步骤 2 和 3，直到目标终点 (Destination vertex) 获得 Final label。
5. **回溯路径 (Tracing back)**: 从终点开始往回找，若相邻两点满足

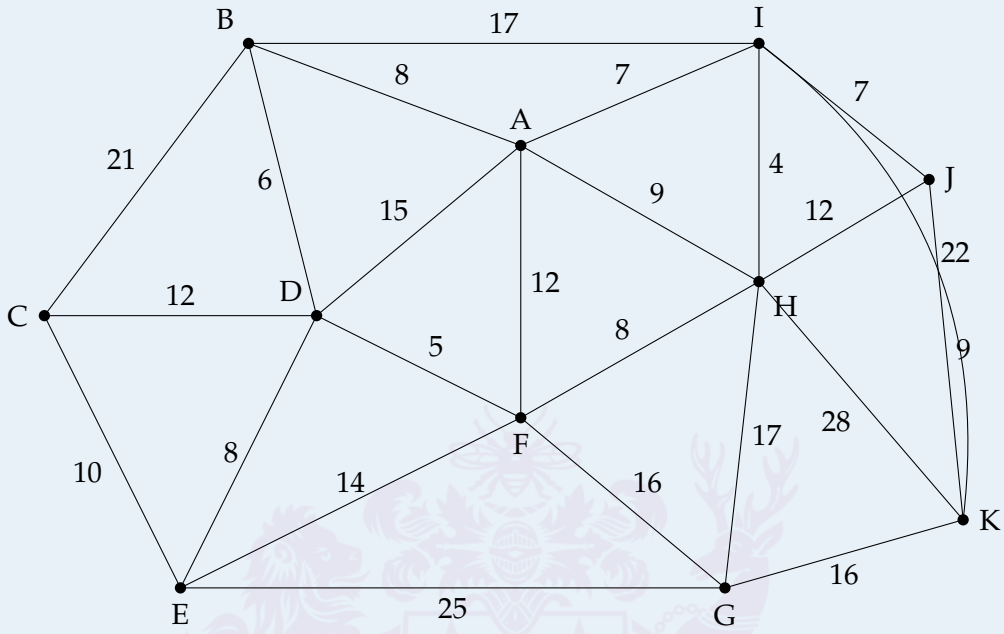
$$\text{Final } \mathbf{B} - \text{Final } \mathbf{A} = \text{Weight of } \mathbf{AB}$$

，则该边 AB 在最短路径上。

Example 3.1 (Paper 1, Jan 2026, Q2 | Dijkstra's Algorithm)

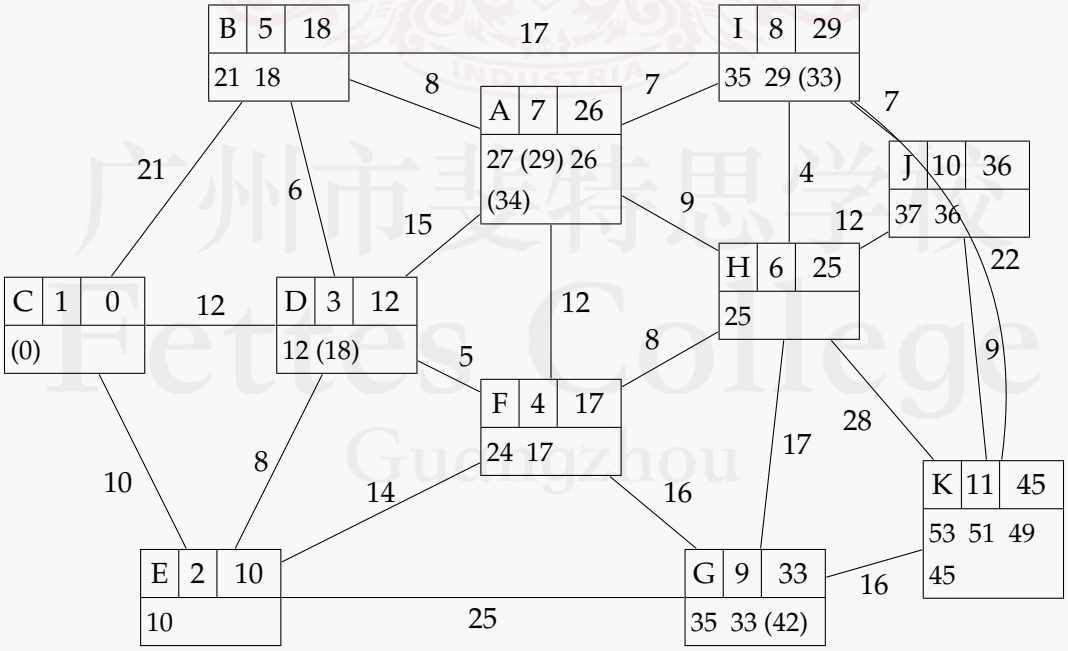
Figure 1 shows a network of paths between eleven viewing points in a bird sanctuary. The number on each arc represents the time taken, in minutes, to walk between the viewing points.

Use Dijkstra's algorithm to determine the quickest route from C to K. You should state the route and the quickest time. (6 Marks)



Solution.

Dijkstra's Algorithm Implementation:



Tracing back the quickest route (回溯最短路径):

从终点 K 的 Final Label (45) 开始往回倒推，寻找满足 Final Y − Final X = weight 的边：

- 45(K) − 36(J) = 9 $\Longrightarrow$ **JK** 在路径上。
- 36(J) − 29(I) = 7 $\Longrightarrow$ **IJ** 在路径上。

- 29(I) – 25(H) = 4 $\Longrightarrow$ **HI** 在路径上。
- 25(H) – 17(F) = 8 $\Longrightarrow$ **FH** 在路径上。
- 17(F) – 12(D) = 5 $\Longrightarrow$ **DF** 在路径上。
- 12(D) – 0(C) = 12 $\Longrightarrow$ **CD** 在路径上。

Quickest Route: C - D - F - H - I - J - K
Quickest Time: 45 minutes

3.2 Route Inspection Algorithm (路线巡检 / 中国邮递员问题)

Key Point 核心知识点 (Eulerian Graphs & Traversability)

- **Eulerian graph (欧拉图)**: 所有顶点的度数 (degree) 均为**偶数 (even)**。包含一条经过所有边且回到起点的 Eulerian circuit（欧拉回路）。最短巡检距离 = 总边权。
- **Semi-Eulerian graph (半欧拉图)**: 恰好有**两个奇数度顶点 (exactly two odd vertices)**。包含一条经过所有边但不回起点的 trail。
- **Route inspection (路线巡检)**: 要求遍历图中的**每一条边至少一次 (every arc at least once)** 并回到起点。若图不是 Eulerian，则必须**重复走 (repeat)** 某些边来“消除”奇点。

Algorithm 算法模型 (Route Inspection Algorithm)

Syllabus 限制：网络最多包含 4 个奇点 (up to four odd nodes)。

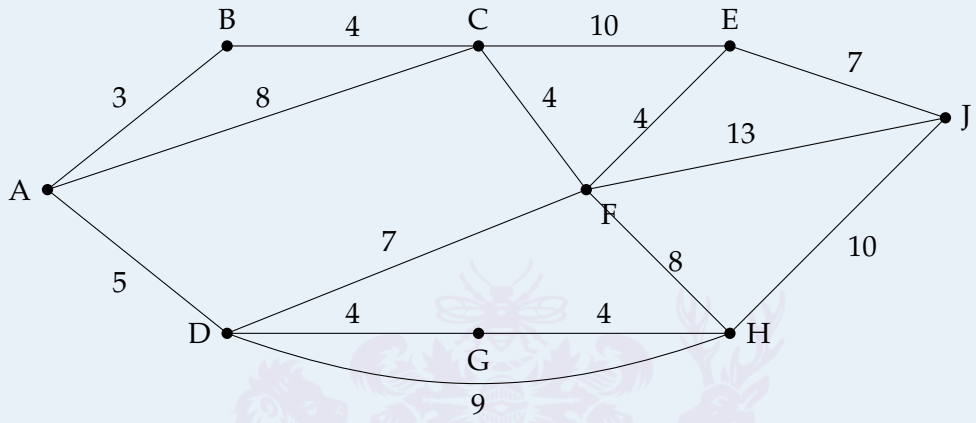
1. **Identify**: 找出图中所有度数为奇数的顶点 (Odd degree vertices)。
2. **Pairings**: 若有 4 个奇点 (A, B, C, D)，写出所有 3 种可能的配对组合：

   • AB + CD • AC + BD • AD + BC

3. **Shortest Path**: 通过观察法或 Dijkstra 算出每对点之间的**最短距离 (Least sum)**。
4. **Select**: 选出总和最小的那个配对方案。
5. **Total Weight**: 将最小配对和加到网络原有的总边权上。即 Total = Sum of all arcs + Least pairing sum。这些被选中的边就是在实际路线中被重复走过的边。

Example 3.2 (Paper 1A, Jan 2026, Q3 | Dijkstra & Route Inspection)

Figure 2 represents a network of pipes in a building. The number on each arc represents the length, in metres, of the corresponding pipe. *[The total weight of the network is 100]*



(a) Use Dijkstra's algorithm to find the shortest path from A to J. State your path and its length. (6 Marks)

On a particular day Kim needs to check each pipe. A route of minimum length, which traverses each pipe at least once and starts and finishes at A, needs to be found.

(b) Use an appropriate algorithm to find the arcs that will need to be traversed twice. You must make your method and working clear. (4 Marks)

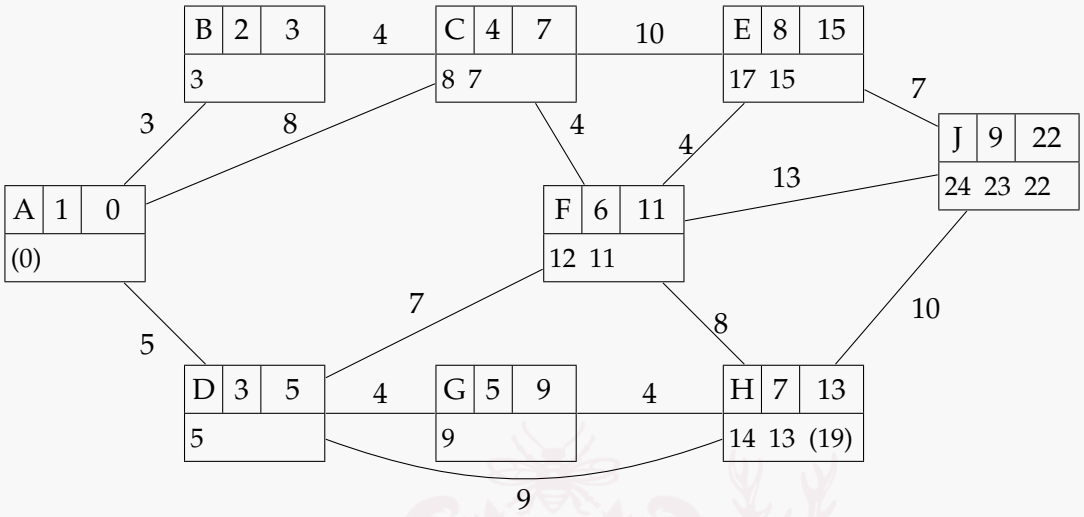
(c) Write down a possible route, giving its length. (2 Marks)

(d) All the pipes directly attached to B are removed. Kim needs to check all the remaining pipes and may now start at any vertex and finish at any vertex. A route is required that excludes all those pipes directly attached to B. State all possible combinations of starting and finishing points so that the length of Kim's route is minimised. State the length of Kim's route. (3 Marks)

Solution.

Part (a): Dijkstra's Algorithm Implementation

广州市第七中学 Fettes College Guangzhou



Shortest route: 通过 Final label 回推，22(J) − 7(E) = 15 $\Longrightarrow$ EJ, 15(E) − 4(F) = 11 $\Longrightarrow$ FE...
依次为 **A - B - C - F - E - J**. **Length: 22** (metres).

Part (b): Route Inspection Algorithm
统计网络中所有顶点的度数，找出 **Odd nodes (奇点)** 为：**A, E, F, J**。
考虑这 4 个奇点的 3 种配对方案，并结合 (a) 问的最短距离计算：

- Pair 1: AF + EJ = 11(via B, C) + 7 = 18 ← **Minimum**
- Pair 2: AE + FJ = 15(via B, C, F) + 11(via E) = 26
- Pair 3: AJ + EF = 22(via B, C, F, E) + 4 = 26

最小重复距离为 18。为了连接 A 和 F，以及 E 和 J，需要被**重复遍历的边 (traversed twice)** 是：**AB, BC, CF, EJ**。

Part (c): Possible Route
Route: **A B A D G H D F H J E J F E C F C B C A** *(答案不唯一，只需包含所有边，且 AB, BC, CF, EJ 走两次即可)*
Length: 100 + 18 = **118** (metres).

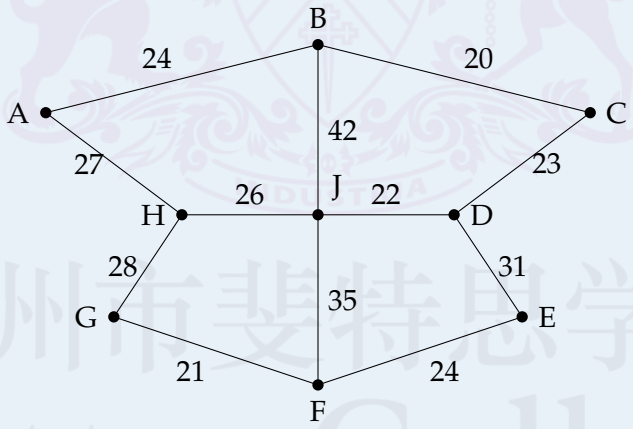
Part (d): Start/Finish Combinations without B (Semi-Eulerian Concept)
逻辑分析：
1. 移除直接连接 B 的管道（即去掉 AB=3 和 BC=4）。
2. **New Total Weight** = 100 − 3 − 4 = 93。
3. 检查移除边后的奇点变化：- 顶点 A 失去 1 条边，度数变为偶数 (Even)。- 顶点 C 失去 1 条边，度数变为奇数 (Odd)。- 顶点 E, F, J 未受影响，仍为奇点 (Odd)。当前图中的 **奇点变为 4 个：C, E, F, J**。
4. 题目允许起点和终点在**不同**的顶点 (Semi-Eulerian trail)，这意味着我们需要留下 **正好 2 个奇点**作为起点和终点。因此，我们需要将另外 2 个奇点配对并重复其最短路径，使其

变成偶点。
5. 为了使总长度最小化，我们必须从 {C, E, F, J} 中找出**最短的单条路径 (Shortest possible pair)**。- CF = 4, EF = 4。这两个是最小的 (相比之下 CE=8, EJ=7 等更长)。
6. **推导结果**：- 若重复 **CF(4)**：C, F 变偶点，剩下 **E, J** 为奇点 $\Longrightarrow$ **Start/Finish at E and J**.
- 若重复 **EF(4)**：E, F 变偶点，剩下 **C, J** 为奇点 $\Longrightarrow$ **Start/Finish at C and J**.
Final Answers:

- **Combinations:** Start/Finish at **E/J**, OR Start/Finish at **C/J**.
- **Minimum Length:** 93(base) + 4(repeated shortest pair) = **97** (metres).

Example 3.3 (Paper 1, Jan 2026, Q3 | Kruskal's Adaptation & Route Inspection)

Figure 2 represents a network connecting nine locations. The number on each arc is the distance in metres between the corresponding locations. *(Total weight 323 metres)*



(i) (a) Explain how Kruskal's algorithm can be adapted to obtain a spanning tree of least weight that contains a specific arc. (1 Mark)
Using the adaptation of Kruskal's algorithm from part (i)(a) on the network in Figure 2,
(b) find the spanning tree of least weight **that includes arc BJ**. You must show your working and clearly state the order in which arcs are added. (3 Marks)
(c) State the weight of the spanning tree found in part (i)(b). (1 Mark)

(ii) An engineer needs to inspect the network in Figure 2. Her route must start and finish at A. She must travel along each arc at least once, minimising the length of her route.
(a) Determine the length of the engineer's route. You must make your method clear and state the arcs which need to be repeated. (5 Marks)
On another day the engineer decides to start her route at F and end at J. She must travel along each arc at least once, minimising the length of her route.

(b) Determine how much shorter this route is than the route obtained in part (ii)(a). You must make your working clear. (3 Marks)

Solution.

Part (i)(a): Adapting Kruskal's Algorithm

Start by initially adding the required specific arc to the tree, and then continue with standard Kruskal's algorithm (adding remaining arcs in ascending order of weight, rejecting any that form a cycle) until the tree is completed.

Part (i)(b): Constructing the Spanning Tree

注：必须列出添加的顺序，并明确指出被拒绝 *(Reject)* 的边。

1. **Add BJ (42)** *(Forced requirement)*
2. **Add BC (20)**
3. **Add FG (21)**
4. **Add DJ (22)**
5. **Reject CD (23)** *(Would form cycle B-C-D-J-B)*
6. **Add AB (24)** 及 **Add EF (24)** *(Either order)*
7. **Add HJ (26)**
8. **Reject AH (27)** *(Would form cycle A-B-J-H-A)*
9. **Add GH (28)**

(9 vertices present, 8 edges added, tree is complete.)

Part (i)(c): Weight of Spanning Tree

Weight = 42 + 20 + 21 + 22 + 24 + 24 + 26 + 28 = **207** (metres).

Part (ii)(a): Route Inspection (Start and Finish at A)

Identify odd nodes in the full network: **B, D, F, H**.

Consider all possible pairings of these 4 odd nodes:

- Pair 1: BD + FH = (20 + 23) + (21 + 28) = 43 + 49 = **92** ← **Minimum**
- Pair 2: BF + DH = (42 + 35) + (22 + 26) = 77 + 48 = 125
- Pair 3: BH + DF = (24 + 27) + (31 + 24) = 51 + 55 = 106

The shortest pairing is BD + FH.

Length of route = Total weight + 92 = 323 + 92 = **415** (metres).

Repeated arcs: To travel BD we repeat **BC, CD**; to travel FH we repeat **FG, GH**.

Part (ii)(b): Route Inspection (Start at F, End at J)

逻辑分析：题目要求起点和终点变为 F 和 J (Semi-Eulerian trail)，这意味着在最终走过的路径中，**只有 F 和 J 必须是奇点**。

- 原图中的奇偶性：F(奇), J(偶), B(奇), D(奇), H(奇)。

广州市菲特思学校 Fettes College Guangzhou

- 目标奇偶性：F(奇), J(奇)，其余全为偶点。
- 结论：F 保持奇点不变；而 **B, D, H, J 这四个点需要改变奇偶性**。因此，我们现在需要对 {B, D, H, J} 进行配对寻找最短重复边。

Consider pairings for B, D, H, and J:

- Pair 1: BJ + DH = 42 + 48 = 90
- Pair 2: BD + HJ = 43 + 26 = **69** ← **Minimum**
- Pair 3: BH + DJ = 51 + 22 = 73

New repeated length is 69.
Difference in length = Old repeated − New repeated = 92 − 69 = **23** (metres).
(The new route is 23 metres shorter.)

3.3 The Travelling Salesman Problem (TSP 旅行商问题)

Key Point 核心知识点 (TSP Basics & Terminology)

- **Tour (回路)**: 一条访问每个顶点且回到起点的 Walk。
- **Classical TSP**: 要求每个顶点**恰好被访问一次 (exactly once)**。
- **Practical TSP**: 允许每个顶点被访问**至少一次 (at least once)**。
- **Triangle Inequality (三角不等式)**: 任意三角形中最长边 ≤ 两短边之和。若全图满足此条件（即 Table of least distances），则 Classical 与 Practical TSP 等价。
- **Bounds (界限)**: TSP 是 NP-hard 问题，故寻找上限 (Upper Bound) 和下限 (Lower Bound) 来“夹逼 (trap)”最优解。最优解一定在 **LB** ≤ Optimal ≤ **UB** 之间。

Algorithm 算法模型 (TSP Upper Bound: Minimum Spanning Tree Method)

用于寻找 Practical TSP 的 Upper Bound (上限)。
1. **Find MST (求最小生成树)**: 使用 Prim 或 Kruskal 算法求出整个网络的最小生成树 (Minimum Spanning Tree)。这确保了所有顶点都被连通。
2. **Double the MST (双倍化生成树)**: 将得到的最小生成树的总权重翻倍 (2×MST weight)。这相当于沿着树的每条边来回走一趟，形成一个必定能遍历所有点并回到起点的初始上限

(Initial upper bound)。

3. **Find Shortcuts (寻找捷径)**: 检查原图中未被包含在 MST 里的边。利用这些边作为“捷径”，跳过已经访问过的顶点，从而避免走重复的回头路。通过捷径不断缩减总距离，得到更优的（更小的）Upper Bound。

Algorithm 算法模型 (TSP Upper Bound: Nearest Neighbour Algorithm)

用于寻找 Upper Bound (上限)。找出的路径一定是合法的解 (possible tour)，但不一定是最优解。

1. 选择指定的起始顶点 (Starting point)。(注：若题目未指定，则需轮流将每个顶点作为起点进行尝试)
2. 走到与当前点距离**最近且未被访问过**的顶点 (Nearest unvisited neighbour)。
3. 重复此操作，直到所有顶点均被访问。
4. 从最后一个顶点直接返回起始顶点，完成 Tour 并计算总长度。
5. **Best Upper Bound (最优上限)**: 若以不同的顶点为起点计算了多个 Tour，则选出其中**总长度最小 (smallest length)** 的那个作为最优上限，以尽可能压缩最优解所在的区间。

Algorithm 算法模型 (TSP Lower Bound: Minimum Spanning Tree Method)

用于寻找 Classical TSP 的 Lower Bound (下限 - Delete-a-node 法)。产生的结果通常不能构成完整的回路 (not a Hamiltonian cycle)。

1. **Delete (删除)**: 从图中移除指定的顶点 X 及其相连的所有边。(注：若要找最佳下限，需轮流删除不同的点进行计算)
2. **RMST (剩余最小生成树)**: 用 Prim 或 Kruskal 算法求剩余网络 (Residual network) 的最小生成树，记下其权重总和。
3. **Reconnect (重连)**: 在刚才被删除的与 X 相连的所有边中，找出**最短且不相同的两条 (two shortest distinct arcs)**，将其权重加回到 RMST 中。

$$\text{Lower Bound} = \text{RMST(excluding } \mathbf{X}\text{)} + \text{Two shortest arcs connecting } \mathbf{X}$$

4. **Best Lower Bound (最优下限)**: 若通过删除不同顶点计算了多个下限，选出其中**值最大 (greatest total)** 的作为最优下限 (Make it as high as possible)。
5. **Optimal Solution (最优解判定)**: 如果你求出的下限恰好构成了一个哈密顿回路 (Hamiltonian cycle)，或者该下限的值与上限相等 (LB = UB)，那么你就找到了**最优解 (optimal solution)**。

Example 3.4 (Paper 1, Jan 2026, Q6 | TSP Upper & Lower Bounds)

The table shows the least distances, in km, between eight towns. Imani wants to visit all eight towns. She needs to find a route which minimises the distance travelled, starting and finishing at A.

	A	B	C	D	E	F	G	H
A	-	24	19	28	25	32	29	31
B	24	-	22	31	25	23	37	41
C	19	22	-	18	21	41	31	35
D	28	31	18	-	24	21	27	26
E	25	25	21	24	-	17	31	25
F	32	23	41	21	17	-	26	31
G	29	37	31	27	31	26	-	24
H	31	41	35	26	25	31	24	-

(a) Show that there are two nearest neighbour routes starting at A. You must make your method clear and state the routes and their lengths. (4 Marks)

(b) State, giving a reason, which route from part (a) gives the better upper bound for the length of Imani's route. (1 Mark)

(c) By deleting A and all of its arcs, use Prim's algorithm, starting at B, to obtain a lower bound for the length of Imani's route. You must show your working. (4 Marks)

A second lower bound of 172 km is obtained by deleting H.

(d) Write down the smallest interval that must contain the optimal length for Imani's route. (1 Mark)

Solution.

Part (a): Upper Bound (Nearest Neighbour Algorithm)

运用 *Algo 3.3*，从 *A* 出发，每次寻找距离当前位置最近且未访问过的点。

1. 从 A 出发，最近的点是 C (19)。路径：A → C。
2. 从 C 出发，最近的未访问点是 D (18)。路径：A → C → D。
3. 从 D 出发，最近的未访问点是 F (21)。路径：A → C → D → F。
4. 从 F 出发，最近的未访问点是 E (17)。路径：A → C → D → F → E。
5. 从 E 出发，寻找未访问点 (B, G, H)。距离为 EB=25, EG=31, EH=25。

出现并列 (Tie)！此时可以选择去 B 或者去 H，因此产生了**两条**不同的路线分支：

- **Branch 1 (选择去 B)**: A → C → D → F → E $\xrightarrow{25}$ **B**。从 B 最近是 G(37)，从 G 到 H(24)，

最后回 A(31)。
Route 1: A - C - D - F - E - B - G - H - A
Length: $19 + 18 + 21 + 17 + 25 + 37 + 24 + 31 =$ **192** (km)

- **Branch 2 (选择去 H)**: A → C → D → F → E $\xrightarrow{25}$ **H**。从 H 最近是 G(24)，从 G 到 B(37)，最后回 A(24)。
Route 2: A - C - D - F - E - H - G - B - A
Length: $19 + 18 + 21 + 17 + 25 + 24 + 37 + 24 =$ **185** (km)

Part (b): Better Upper Bound
Route 2 (A-C-D-F-E-H-G-B-A) gives the better upper bound, **because 185 is the shorter length** ($185 < 192$). （最优上限应当尽可能小）

Part (c): Lower Bound (Delete-a-node Method)
运用 *Algo 3.3*，删除节点 *A*。
Step 1: Prim's Algorithm (Starting at B)

- Add **BC** (22)
- Add **CD** (18)
- Add **DF** (21) 或 *CE (21)* 均可，这里以 *DF* 为例
- Add **FE** (17)
- Add **EH** (25)
- Add **HG** (24)

Weight of RMST (Residual Minimum Spanning Tree) $= 22 + 18 + 21 + 17 + 25 + 24 =$ **127**。
Step 2: Reconnect node A
检查矩阵中 A 行的值，找到连接 A 的两条最短的边：**AC (19)** 和 **AB (24)**。
Lower Bound = RMST + Two shortest arcs = $127 + 19 + 24 =$ **170** (km)。

Part (d): Smallest Interval
已知两个 Lower Bounds: 170 (from part c) 和 172 (given)。**Best Lower Bound 选最大：172**。
已知两个 Upper Bounds: 192 和 185 (from part a)。**Best Upper Bound 选最小：185**。
最优解 d 必被夹在最佳上下限之间：

$$172 \leq d \leq 185$$

第 4 章 Critical Path Analysis (关键路径分析)

Key Point 核心知识点 (Activity Networks, Gantt Charts & Scheduling)

1. 网络基础与定义 (Network Basics):

- **Precedence table**: 显示活动间的依赖关系。
- **Activity on arc**: **弧 (Arcs)** 代表活动及其耗时，**节点 (Nodes)** 代表事件。起点为 **Source node** (通常从 0 编号)，终点为 **Sink node**。
- **Dummy Activities (虚活动)**: 不消耗时间和资源。作用有二：1) 表达严格的依赖逻辑 (**Logic**)；2) 确保两节点间最多只有一条弧，使每个活动由唯一的起始与结束节点组合表示 (**Uniqueness**)。

2. 时间推导与关键路径 (Event Times & Critical Path):

- **Early Event Time**: 顺向推导 (**Forward pass**)，遇多条路径交汇取 **最大值 (Maximum)**。
- **Late Event Time**: 逆向推导 (**Backward pass**)，遇多条路径交汇取 **最小值 (Minimum)**。
- **Critical Path (关键路径)**: 从 Source 到 Sink **最长的路径**。路径上节点的 Early time = Late time。
- **易错考点**: 连接两个关键事件的活动 **不一定**是关键活动 (Point 8)。
- **Total Float (总浮动时间)**: Latest finish time − Earliest start time − Duration。关键活动的 Total Float 必为 0。

3. 甘特图 (Gantt / Cascade Chart):

- 用图形直观表示所有活动的可行时间范围。通常每个活动从其**最早开始时间**画起，实线框代表持续时间 (Duration)，虚线代表可用的浮动时间 (Float)。

4. 排班规则与工人数量 (Scheduling & Workers):

- **Lower Bound for Workers**: LB = $\frac{\text{Sum of all activity times}}{\text{Critical time of the project}}$，结果必须 **向上取整** (smallest integer ≥ value)。
- **排班约束**: 1) 工人一次只能做一项活动；2) 活动一旦开始必须 **不间断地完成**。
- **排班优先级 (Scheduling Rules)**: 1) 始终优先分配给 **当前最早空闲 (first available)** 的工人。2) 若一个工人有多个活动可选，优先分配 **Late time (最迟结束时间) 最小**的活动。

广州市菲特思学校 Fettes College Guangzhou

Example 4.1 (Paper 1A, Jan 2026, Q4 | Drawing Network from Table)

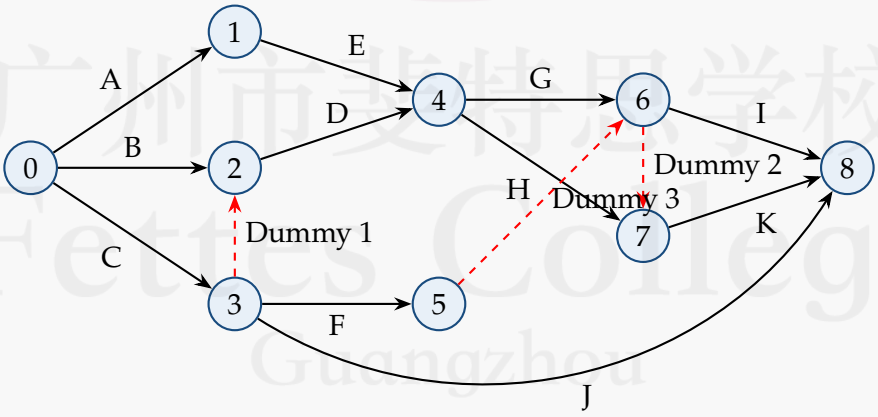
Draw the activity network described in this precedence table, using activity on arc and dummies only where necessary. (5 Marks)

Activity	Immediately preceding	Activity	Immediately preceding
A	–	G	D, E
B	–	H	D, E
C	–	I	F, G
D	B, C	J	C
E	A	K	G, H
F	C		

Solution.

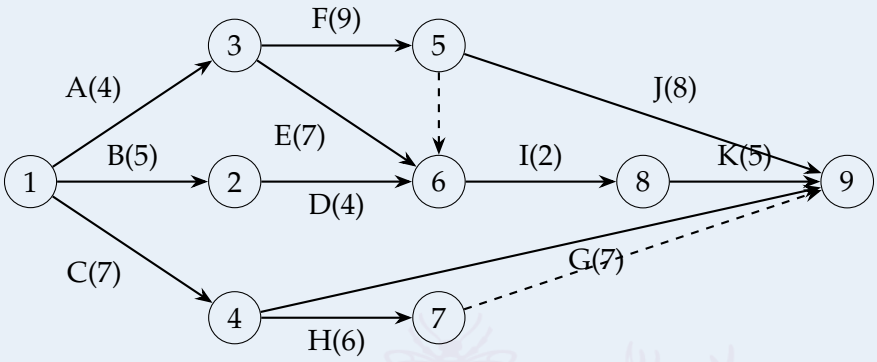
绘制网络图时必须注意：

1. **唯一性 (Uniqueness)**: G 和 H 都有完全相同的依赖 (D, E)，因此必须在它们的终点之间引入一个 Dummy，防止它们共享完全相同的起始点和终点节点。
2. **逻辑分离 (Logic)**: D 依赖 B 和 C，而 F, J 仅依赖 C。因此 C 不能直接接入 D 的起点节点，需要引入 Dummy 把 C 的完成状态“单向传递”给 D。



Example 4.2 (Paper 1A, Jan 2026, Q6 | Event Times, Gantt Chart & Scheduling)

A project is modelled by the activity network shown below.



(a) Complete the diagram to show the early event times and the late event times. (4 Marks)

(b) State the minimum project completion time. (1 Mark)

(c) Calculate a lower bound for the minimum number of workers required. (2 Marks)

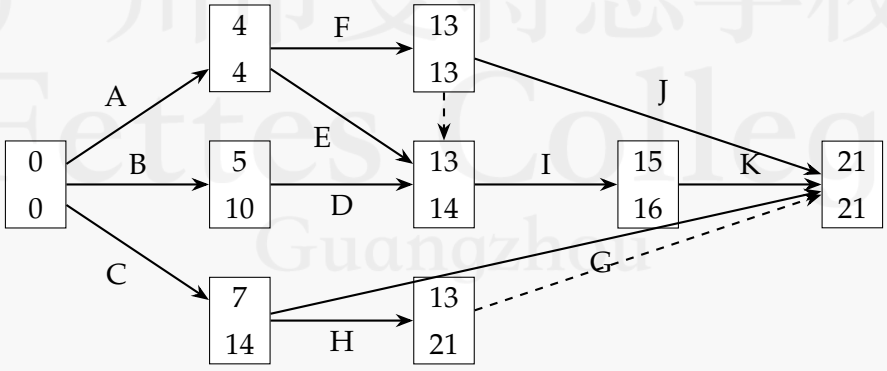
(d) Draw a cascade (Gantt) chart for this project. (4 Marks)

(e) Construct a scheduling diagram to show that this project can be completed with three workers in just one more hour than the minimum project completion time. (3 Marks)

Solution.

Part (a): Event Times Calculation

注：原题网络图连线有误，基于官方 *Mark Scheme*，*G(7)* 实际由节点 4 连向节点 9。下表为修正后的时间推导：



Part (b) & (c): Time and Bounds

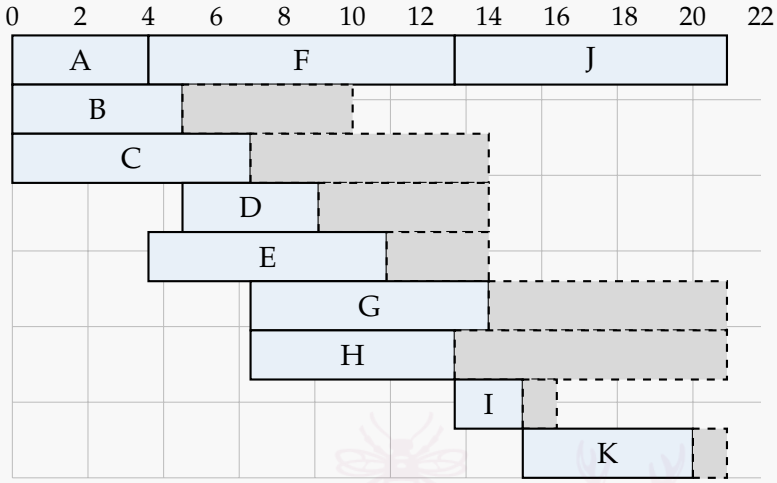
(b) Minimum project completion time = **21 hours**.

(c) Sum of activities = $4 + 5 + 7 + 9 + 7 + 4 + 7 + 6 + 8 + 2 + 5 = 64$.

LB = $\frac{64}{21} \approx 3.048 \implies$ **At least 4 workers required**.

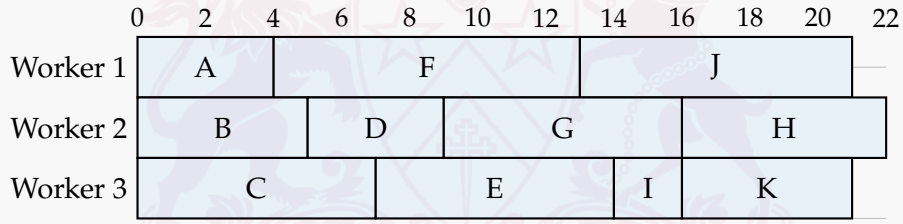
Part (d): Gantt (Cascade) Chart

实线白底矩形代表持续时间 (Duration)，阴影虚线矩形代表浮动时间 (Float)。通过绝对坐标彻底居中了字母。

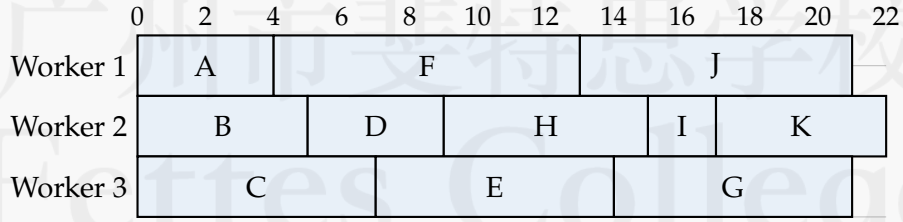


Part (e): Scheduling Diagram (3 workers in 22 hours)

Standard Schedule : Worker 1 处理关键路径 A, F, J。



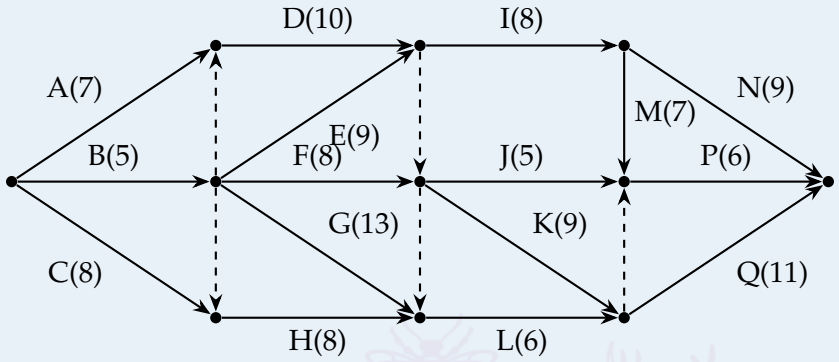
Alternative Schedule (有效备选方案): 利用放宽的 22 小时限度，H、I、G 等活动的顺序可以灵活对调。



广州市菲特恩学校 Fettes College Guangzhou

Example 4.3 (Paper 1, Jan 2026, Q7 | Full Analysis & Scheduling)

A project is modelled by the activity network below.



(a) Complete the table to show the immediately preceding activities. (3 Marks)

(b) Complete the diagram to show early and late event times. (4 Marks)

(c) State the critical activities. (1 Mark)

(d) Calculate a lower bound for the minimum number of workers required. (2 Marks)

(e) Draw a schedule to show how the activities can be completed in the minimum time, using the minimum number of workers. (4 Marks)

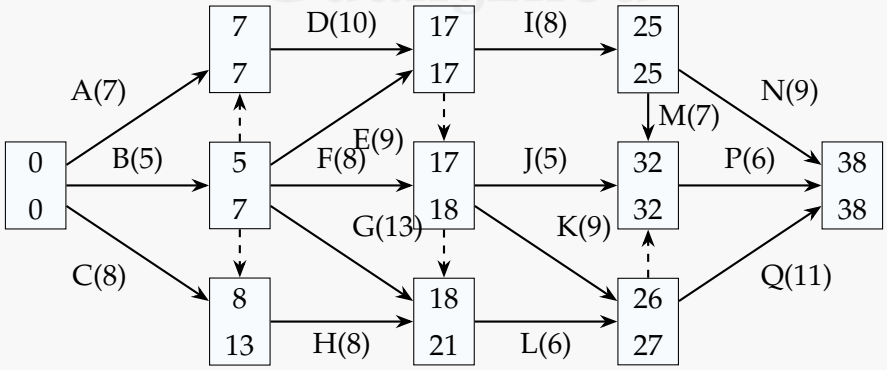
Solution.

Part (a): Precedence Table

通过观察图中箭头的指向以及虚线 (Dummy) 的传导逻辑：

Act.	Preceding	Act.	Preceding	Act.	Preceding	Act.	Preceding
A	–	E	B	I	D, E	M	I
B	–	F	B	J	D, E, F	N	I
C	–	G	B	K	D, E, F	P	J, K, L, M
D	A, B	H	B, C	L	G, H	Q	K, L

Part (b) & (c): Event Times and Critical Activities



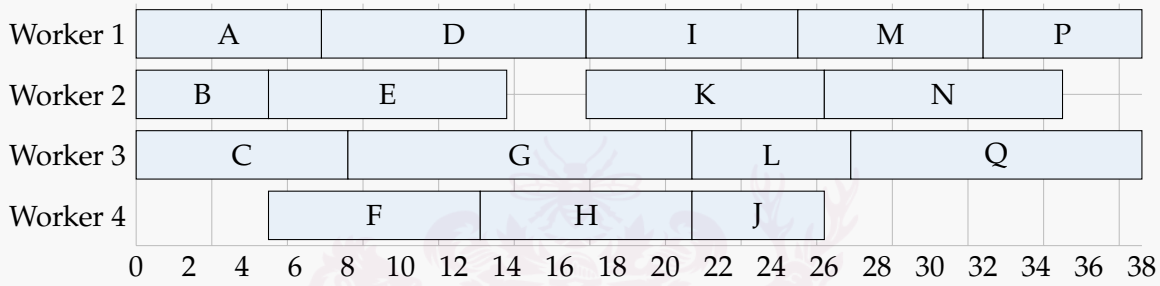
(c) 计算 Float = 0 的活动，唯一的关键路径为: **A, D, I, M, P**。

Part (d): Lower Bound for Workers

Total Sum = $7 + 5 + 8 + 10 + 9 + 8 + 13 + 8 + 8 + 5 + 9 + 6 + 7 + 9 + 6 + 11 = 129$.

LB = $\frac{129}{38} \approx 3.39 \implies$ **4 workers**.

Part (e): Scheduling Diagram (4 workers, 38 hours)



广州市斐特思学校
Fettes College
Guangzhou

第 5 章 Linear Programming (线性规划)

Key Point 核心知识点 (Linear Programming Formulation & Solutions)

1. 建立模型 (Formulation):

- **Decision variables (决策变量)**: 明确定义变量，如令 x, y, z 为对应物品的数量。
- **Objective function (目标函数)**: 明确是求 **Maximise** (最大化，如利润) 还是 **Minimise** (最小化，如成本)，并写出代数表达式。
- **Constraints (约束条件)**: 将题目的资源限制、比例限制等转化为不等式 (Inequalities)。

2. 求解方法 (Solving Methods):

- **Feasible region (可行域)**: 图中满足所有约束不等式的区域。考试中通常要求给**不满足 (not included)** 的区域打阴影，留白可行域并标记为 R。
- **Objective line method (目标线法/平移法)**:
  - **Maximum point**: 目标线平行移动，**离开 (leaves)** 可行域时覆盖的最后一个点。
  - **Minimum point**: 目标线平行移动，**进入 (enters)** 可行域时覆盖的第一个点。
- **Vertex method (顶点法)**:
  - 求出可行域所有顶点的精确坐标 (联立解方程组)。
  - 将各点代入目标函数 (evaluate the objective function)。
  - 选择使目标函数达到最优值的顶点。

3. 整数解 (Integer Solutions): 如果实际问题要求解必须是整数(如生产桌子的数量)，且算出的最优顶点包含小数，则需要在最优顶点附近测试位于可行域内的 **整数坐标点 (integer coordinates)**，比较得出整数条件下的最优解。

广州市斐特思学校 Fettes College Guangzhou

Example 5.1 (Paper 1, Jan 2026, Q5 | LP Formulation (3 Variables))

A carpenter produces three sizes of table, small, medium and large. The information below shows the quantity of wood required, production time, profit and total available resources.

	Quantity of wood	Production time	Profit
Small	1.2 m^2	4 hours 20 minutes	£275
Medium	2.5 m^2	7 hours 30 minutes	£425
Large	4.0 m^2	9 hours	£650
Total available	60 m^2	200 hours	

The tables produced should meet the following conditions:

- at least 40% of the tables must be small
- there must be at least twice as many small tables as medium tables
- no more than 30% of the tables must be large

The carpenter wishes to maximise the profit from the sale of the tables. Let x, y, and z be the number of small, medium, and large tables produced. Formulate this as a linear programming problem. You should clearly state the objective function and give the constraints as simplified inequalities with integer coefficients. (8 Marks)

Solution.

1. Objective Function (目标函数):

Maximise $P = 275x + 425y + 650z$

2. Resource Constraints (资源约束):

Wood: $1.2x + 2.5y + 4.0z \leq 60 \implies \mathbf{12x + 25y + 40z \leq 600}$

Time (统一转化为小时或分钟，此处用小时):

4 h 20 min = $\frac{13}{3}$ h; 7 h 30 min = $\frac{15}{2}$ h

$\frac{13}{3}x + \frac{15}{2}y + 9z \leq 200$ (两边同乘 6 化整) $\implies \mathbf{26x + 45y + 54z \leq 1200}$

3. Ratio Constraints (比例约束):

At least 40% small:

$x \geq 0.4(x + y + z) \implies 10x \geq 4x + 4y + 4z \implies 6x \geq 4y + 4z \implies \mathbf{3x \geq 2y + 2z}$

At least twice as many small as medium:

$\mathbf{x \geq 2y}$

No more than 30% large:

$z \leq 0.3(x + y + z) \implies 10z \leq 3x + 3y + 3z \implies \mathbf{7z \leq 3x + 3y}$

广州市斐特思学校 Fetes College @Guangzhou

4. Non-negativity Constraint (非负约束):

$\mathbf{x, y, z \geq 0}$

Example 5.2 (Paper 1, Jan 2026, Q8 | Graphical, Vertex & Integer Solutions)

A linear programming problem is formulated as follows:

Maximise $P = 6x + 5y$

Subject to: $y \leq 3x + 10$ $x + 4y \leq 162$ $5x + 2y \leq 200$ $y \geq 15$ $x \geq 0$

(a) Complete the graph in the answer book to show these inequalities, shading those regions which are **not** included in the feasible region. Label the feasible region R. (3 Marks)

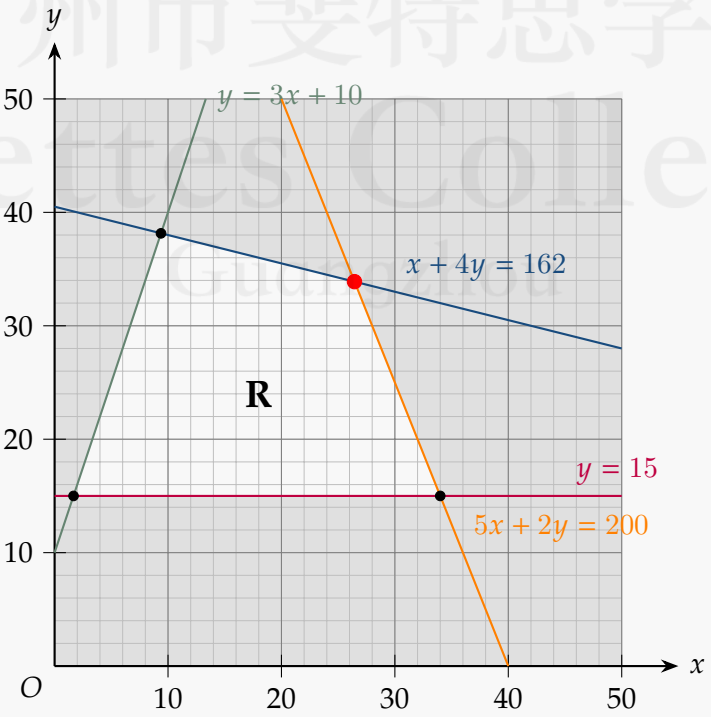
(b) Use the **vertex method** to determine the exact coordinates of the optimal vertex. (4 Marks)

(c) The practical solution requires **integer solutions**. Determine the optimal solution and state the value of the objective function. (3 Marks)

(d) The objective function is now changed to $P = kx + y$. Determine the **range of values for** k for which the optimal vertex is unchanged. (2 Marks)

Solution.

Part (a): Graphical Completion



Part (b): Vertex Method

求解包围可行域 R 的四个顶点的精确坐标，并代入 $P = 6x + 5y$ 测试：

- 交点 1 ($y = 15, y = 3x + 10$): $x = 5/3 \implies (5/3, 15) \implies P = 6(5/3) + 5(15) = 85$
- 交点 2 ($y = 3x+10, x+4y = 162$): $13x = 122 \implies (\frac{122}{13}, \frac{496}{13}) \implies P = \frac{3212}{13} \approx 247.1$
- 交点 3 ($x + 4y = 162, 5x + 2y = 200$): $9x = 238 \implies (\mathbf{\frac{238}{9}}, \mathbf{\frac{305}{9}}) \implies P = \frac{2953}{9} \approx \mathbf{328.1}$
- 交点 4 ($5x+2y = 200, y = 15$): $5x = 170 \implies (34, 15) \implies P = 6(34)+5(15) = 279$

最大值产生在交点 3，因此最优顶点坐标为 $(\mathbf{\frac{238}{9}}, \mathbf{\frac{305}{9}})$.

Part (c): Integer Solutions

最优非整数点为 $(\frac{238}{9}, \frac{305}{9}) \approx (26.44, 33.89)$。我们在其周围测试可行域内的整数坐标点：

- 尝试 $(26, 33)$: 满足所有约束，$P = 6(26) + 5(33) = 156 + 165 = 321$
- 尝试 $\mathbf{(26, 34)}$: 满足所有约束 $(x + 4y = 162 \leq 162, 5x + 2y = 198 \leq 200)$, $\mathbf{P = 6(26) + 5(34) = 326}$
- 尝试 $(27, 33)$: 不满足约束 $(5x + 2y = 135 + 66 = 201 > 200)$，拒绝。

最优整数解为 $\mathbf{x = 26, y = 34}$，目标函数值为 $\mathbf{P = 326}$。

Part (d): Range of values for k

目标函数平行线可以写为 $y = -kx + P$。要使最优顶点 $(\frac{238}{9}, \frac{305}{9})$ 保持不变，目标线的斜率 $-k$ 必须夹在交于该点的两条约束线的斜率之间。这两条约束线为：1) $x + 4y = 162 \implies y = -\frac{1}{4}x + 40.5$ (Gradient $= -\frac{1}{4}$)

2) $5x + 2y = 200 \implies y = -\frac{5}{2}x + 100$ (Gradient $= -\frac{5}{2}$)

因为目标线斜率需要比 $-\frac{1}{4}$ 更陡，且比 $-\frac{5}{2}$ 更缓（考虑负数大小关系）：

$$-\frac{5}{2} < -k < -\frac{1}{4}$$

两边同乘 -1 并变号，得到：

$$\mathbf{\frac{1}{4} < k < \frac{5}{2}}$$

A Appendix 1: Glossary (核心专业词汇总结)

本词汇表已按考点主题分类，涵盖从最基础到高阶的所有专业术语。

English Term	中文释义	Explanation / Example in Context
1. Graph Basics (图论基础概念)		
Vertex / Node	顶点 / 节点	图中的点，代表位置、事件或对象。
Edge / Arc	边 / 弧	连接两个顶点的线。带有箭头的称为 Directed arc (有向弧)。
Degree / Valency / Order	度数	连接到某个顶点的边的数量。分 Even degree (偶数度) 和 Odd degree (奇数度)。
Walk	路线	沿着边从一个顶点到另一个顶点的连续序列。
Path	路径	任何**顶点 (vertex)** 都不被访问超过一次的 Walk。
Trail	轨迹	任何**边 (edge)** 都不被访问超过一次的 Walk。
Cycle	环 / 回路	起点和终点相同，且除此之外不重复访问任何其他顶点的 Walk。
Tree	树	连通且**无环 (no cycles)** 的图。
Spanning tree	生成树	包含原图所有顶点，且本身是一棵树的连通子图。
Complete graph	完全图	图中每一个顶点都与其他所有顶点直接相连。
Bipartite graph	二分图	顶点可分成两组，所有的边只连接两组之间的顶点，同组内无边。
Isomorphic graphs	同构图	图的形状画法不同，但顶点的连接逻辑和拓扑结构完全一致。
Adjacency matrix	邻接矩阵	矩阵内的数字代表两顶点之间连接的**边的数量**。
Distance matrix	距离矩阵	矩阵内的数字代表两顶点之间边的**权重 (weight)**。
2. Route Inspection & TSP (路线巡检与旅行商问题)		
Eulerian graph	欧拉图	所有顶点的度数均为**偶数**，包含一条经过所有边且回到起点的 Eulerian cycle。
Semi-Eulerian graph	半欧拉图	恰好有**两个奇数度顶点**的图，包含一条经过所有边但不回起点的 Trail。
Hamiltonian cycle	哈密顿回路	(TSP) 经过图中**每一个顶点**恰好一次，并回到起点的 Cycle。

English Term	中文释义	Explanation / Example in Context
Tour	巡回路线	(TSP) 一条访问每个顶点（可重复）并且回到起点的路线。
Classical TSP	经典旅行商问题	要求每个顶点**恰好 (exactly)** 被访问一次的最小路径问题。
Practical TSP	实际旅行商问题	允许每个顶点被访问**至少 (at least)** 一次的最小路径问题。
Triangle inequality	三角不等式	任意三角形中最长边 ≤ 两短边之和。若全图满足此条件，则 Classical 与 Practical TSP 等价。
3. Matchings (匹配)		
Matching	匹配	二分图中任意两条边都没有公共顶点的边集合。
Alternating path	交替路径	起点和终点均为未匹配顶点，且“匹配边”和“未匹配边”交替出现的路径。
4. Activity Networks (活动网络)		
Precedence table	前置活动表	显示项目中各项活动依赖关系的表格。
Source / Sink node	起点 / 终点	整个活动网络的唯一出发节点（通常编号 0）和唯一结束节点。
Dummy activity	虚活动	用虚线表示。不耗时、不耗资源，仅用于表达逻辑依赖关系与确保标识唯一性。
Critical path	关键路径	从 Source 到 Sink 最长的路径，该路径上节点的 Early time = Late time。
Total Float	总浮动时间	Late Finish − Early Start − Duration，关键活动的 Float 必为 0。
5. Linear Programming & Others (线性规划与其他)		
Objective function	目标函数	(LP) 需要被最大化 (Maximise) 或最小化 (Minimise) 的表达式。
Feasible region	可行域	(LP) 满足所有约束不等式的点集区域（常在图中留白并标为 R）。
Pivot	基准点 / 中点	(Binary Search) 每次取中间位置 $\lceil \frac{n+1}{2} \rceil$ 的元素用于与目标值比较。

B Appendix 2: Algorithm Index (算法速查索引)

Summary of Algorithms Required for D1

1. Sorting, Searching & Packing

Binary Search (二分查找) 1
Bubble Sort (冒泡排序) 2
Quick Sort (快速排序) 4
Bin-Packing (三种装箱算法) 5

2. Minimum Spanning Trees (MST)

Kruskal's Algorithm (全局贪心) 10
Prim's Algorithm on a Graph 10
Prim's Algorithm on a Matrix 11

3. Shortest Paths & Routes

Dijkstra's Algorithm (最短路径) 15
Route Inspection (中国邮递员) 17

4. Travelling Salesman (TSP)

Upper Bound: MST + Shortcuts 22
Upper Bound: Nearest Neighbour 23
Lower Bound: Delete-a-node 23

5. Activity Networks (CPA)

Forward & Backward Pass 26
Total Float Calculation 26
Workers Scheduling Rules 26

6. Linear Programming (LP)

Objective Line Method (平移法) 32
Vertex Method (顶点测试法) 32
Integer Solutions Search (整数解) 32

广州市斐特思学校
Fettes College
Guangzhou