Decision Mathematics 1

Edexcel IAL D1 (WDM01/01); the discrete option completing the IAL Mathematics unit table

Decision Mathematics 1: algorithms, networks and linear programming

- carry out sorting and **bin packing** algorithms;
- find minimum spanning trees and shortest paths in networks;
- solve route inspection and travelling salesman problems;

Decision Mathematics 1: algorithms, networks and linear programming (2)

- analyse projects with **critical path** networks;
- formulate and solve linear programming problems.

Sorting and packing algorithms

- Pass 1: swap 5 and 1, swap 5 and 4, swap 5 and 2, leave 5 and 8: the list is 1, 4, 2, 5, 8.
- Pass 2: leave 1 and 4, swap 4 and 2, leave 4 and 5: the list is 1, 2, 4, 5, 8.
- Pass 3: no swaps, so the list is sorted.

Sorting and packing algorithms (2)

- The middle item (4th of 7) is 5: 2, 3, 4, **5**, 9, 7, 6.
- Sublist 2, 3, 4 has pivot 3, and is already in order. Sublist 9, 7, 6 has pivot 7: 6, **7**, 9.
- The sorted list is 2, 3, 4, 5, 6, 7, 9.

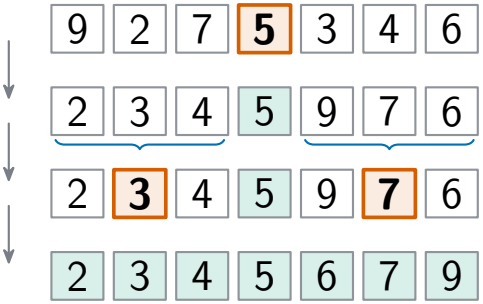
Sorting and packing algorithms (3)

- First-fit: take the items in the order given, and put each in the first bin with room.
- First-fit decreasing: sort the items into descending order first, then use first-fit.
- Full-bin: look for combinations that fill a bin exactly, then use first-fit for the rest.

Sorting and packing algorithms (4)

- Lower bound: $\frac{27}{10} = 2.7$, so at least 3 bins.
- First-fit: 7 $\rightarrow$ bin 1; 5 $\rightarrow$ bin 2 (no room in bin 1); 4 $\rightarrow$ bin 2; 3 $\rightarrow$ bin 1; 6 $\rightarrow$ bin 3; 2 $\rightarrow$ bin 3. Bins: (7, 3), (5, 4), (6, 2).
- First-fit decreasing, with 7, 6, 5, 4, 3, 2: bins (7, 3), (6, 4), (5, 2).
- Both use 3 bins, which equals the lower bound, so they are optimal.

Figure: Sorting and packing algorithms



- **Quick sort** with the middle-item pivot: quote the pivot and the list after every partition
- Match each part of the diagram to the text.

Minimum spanning trees

- **Kruskal's algorithm 克鲁斯卡尔算法**: list the edges in order of weight. Add the smallest edge that does not make a cycle, and repeat until every vertex is connected.
- **Prim's algorithm 普里姆算法**: start at any vertex. Repeatedly add the smallest edge that joins a vertex in the tree to a vertex not yet in the tree.
- Kruskal: CD 1, AC 2, BC 3; then AB 4 and BD 5 are rejected because they would make cycles.

Minimum spanning trees (2)

- Prim from A: AC 2, then CD 1, then CB 3.
- Both give the MST AC, CD, CB, with total weight 6.

Shortest paths

- A $= 0$, permanent. Temporary labels: B 4, C 2.
- C $= 2$, permanent. From C: B would be $2 + 3 = 5$, larger than 4, so B stays 4; D becomes $2 + 1 = 3$.
- D $= 3$, permanent. From D: B would be $3 + 5 = 8$, so B stays 4.

Shortest paths (2)

- B = 4, permanent.
- The shortest route from A to D is A–C–D, of length 3; from A to B it is the direct edge, of length 4.

The Chinese postman problem

- If every vertex has even degree, the shortest route is the total weight of the edges.
- Otherwise, list the odd vertices. Find every way to pair them up, and the shortest distance for each pair. Choose the pairing with the smallest total, and repeat those edges.
- Shortest route = total weight of all edges + the smallest pairing total.

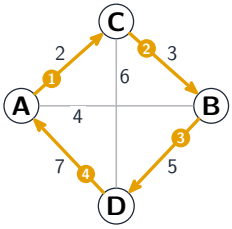
The travelling salesman problem

- **Upper bound**: an actual route. When the distances satisfy the triangle inequality, doubling the weight of the MST gives an upper bound; shortcuts can improve it. The **nearest neighbour algorithm 最近邻算法** also gives one: from the start, go to the nearest unvisited vertex each time, then return to the start.
- **Lower bound**: delete one vertex, find the MST of the rest, and add the two smallest edges from the deleted vertex.
- Nearest neighbour from A: A–C (2), C–B (3), B–D (5), D–A (7). Total 17, an upper bound.

The travelling salesman problem (2)

- The MST is AC, BC, BD, with total 10, so doubling gives the weaker upper bound 20.
- Lower bound, deleting A: the MST of B, C, D is BC + BD = 8, and the two smallest edges from A are AC + AB = 6. So the lower bound is 14.
- So $14 \leq$ optimal route ≤ 17. With only four vertices, we can check all three routes: A–B–C–D–A = 20, A–B–D–C–A = 17 and A–C–B–D–A = 17. The optimal route has length 17.

Figure: The travelling salesman problem



nearest neighbour from A:
A–C–B–D–A $= 2+3+5+7 = 17$
upper bound
delete A: MST of B, C, D $= 3+5 = 8$
plus AC + AB $= 2+4$:
lower bound 14
$14 \leq$ optimal tour ≤ 17

- Nearest neighbour gives an upper bound, the deleted-vertex MST a lower bound: the optimal tour lies between 14 and 17
- Match each part of the diagram to the text.

exercises.lol exercises.com.cn

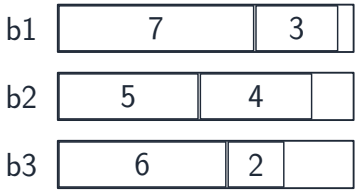
Critical path analysis

- **Earliest event times 最早事件时间**: work forwards from 0. At each event, take the largest of (earliest time of the previous event + duration).
- **Latest event times 最晚事件时间**: work backwards from the end. At each event, take the smallest of (latest time of the next event − duration).
- **Total float 总时差** of an activity = latest finish time − earliest start time − duration.

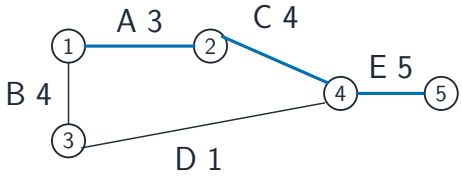
Critical path analysis (2)

- A **critical path** 关键路径 is a path of activities with zero **total float**. Its length is the shortest time in which the project can be completed.
- Earliest times: A ends at 3, B at 4, C at 7, D at 5. E starts at $\max(7, 5) = 7$ and ends at 12.
- The **critical path** is A–C–E, and the project takes 12 days.
- D must finish by 7, so its total float is $7 - 4 - 1 = 2$ days. B must finish by 6, so its float is $6 - 0 - 4 = 2$ days.

Figure: Critical path analysis



first-fit: (7,3), (5,4), (6,2) — three bins, none over capacity



critical path A–C–E = 12 days; D has float 2

- Left: the first-fit bins of worked example 3. Right: the activity network of worked example 8, with the **critical path** A–C–E highlighted.
- Match each part of the diagram to the text.

Linear programming

- the **ruler method 尺子法**: draw a line of the objective and slide a ruler parallel to it; the last point of the feasible region it touches is optimal;
- the **vertex method 顶点法**: work out the objective at each vertex of the feasible region and choose the best.
- The lines $x + y = 10$ and $2x + y = 14$ meet where $x = 4$, $y = 6$.

Linear programming (2)

- The vertices of the feasible region are $(0, 0)$, $(7, 0)$, $(4, 6)$ and $(0, 10)$, giving $P = 0$, 21, 24 and 20.
- The maximum is $P = 24$, at $x = 4$, $y = 6$.

Sources for this lesson

- Pearson Edexcel, *International Advanced Level Mathematics, Further Mathematics and Pure Mathematics specification* (2018), unit D1.
- E. W. Dijkstra, "A note on two problems in connexion with graphs", *Numerische Mathematik*, 1 (1959).
- Worked examples written for this lesson in the style of WDM01 questions.

Common errors

- Not showing the list after each pass of a sort.
- In a **quick sort**, choosing the first item as the pivot instead of the middle one.
- In **Kruskal's algorithm**, adding an edge that makes a cycle.

Common errors (2)

- In **Dijkstra's algorithm**, not recording the order in which labels become permanent.
- In route inspection, giving only one pairing of the odd vertices instead of all of them.
- Calling the nearest neighbour route the optimal route; it is only an upper bound.

Common errors (3)

- Forgetting a **dummy**, or forgetting the non-negativity constraints.
- Shading the feasible region instead of the unwanted region, when the question says to shade what is not wanted.

Key terms

- **Minimum spanning tree**: a tree connecting all the vertices with the least total weight.
- **Dijkstra's algorithm**: an algorithm for the shortest path from one vertex to the others.
- **Route inspection problem**: finding the shortest closed route that uses every edge.

Key terms (2)

- **Critical path**: the longest path through an activity network; its activities have zero float.
- **Total float**: the time by which an activity can be delayed without delaying the project.
- **Feasible region**: the set of points that satisfy all the constraints.

Check yourself

- 1.
- How many passes does a **bubble sort** of 4, 3, 2, 1 need, including the final pass with no swaps? 2.
- Items of sizes 6, 6, 5, 4, 3 go into bins of size 10.
- Find the lower bound for the number of bins.