Chapter 3

① How many topological orderings does it have?

6.

② Explain how you get computed it.

Find a node $v$ with no incoming edges and order it first,

Delete the node $v$ from the graph $G$., add $v$ into the order T.

Recursively compute the rest of $G'$ whose node set is $G-\{v\}$.

and append the following deleted node after previous nodes in T.

③ List the topological order.

$a \to b \to c \to d \to e \to f$

$a \to b \to d \to c \to e \to f$

$a \to b \to d \to e \to c \to f$

$a \to d \to b \to c \to e \to f$

$a \to d \to b \to e \to c \to f$

$a \to d \to e \to b \to c \to f$

# Chapter 4 : Scheduling

I design 4 possible handling ideas.

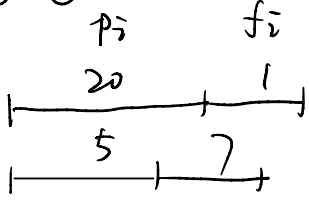
① sort the jobs in a descending order of $p_i + f_i$

② sort the jobs in an ascending order of $p_i + f_i$

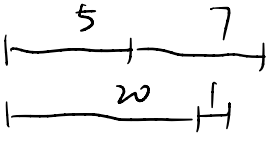
③ sort the jobs in a descending order of $f_i$ (right!!)

④ sort the jobs in an ascending order of $f_i$

① ④ :



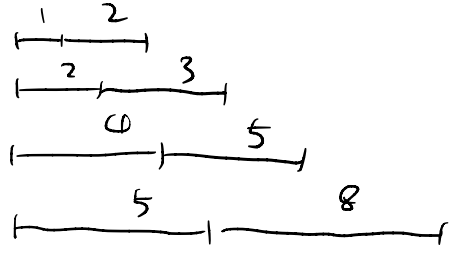

20+5+7 = 32 (not optimal)



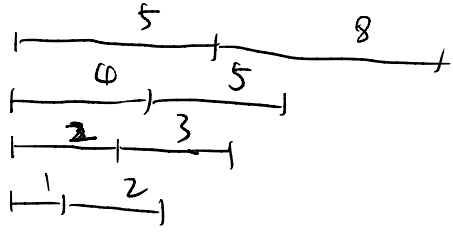

5+20+1 = 26 (optimal)

① and ④ are false.

②




1+2+4+5+8 = 20 (not optimal)




5+4+2+1+2

= 14 (optimal)

② is false.

a)
Algorithm:

Input: A list of jobs with the index (0, ···· n-1)

Use the quick sort to sort all the jobs in an descending order of J[i].f.

From index i = 0 ···· n-1, feed a job to the super-computer wait for the job to finish pre-processing and send the job to any PC.

b) the running time is O(nlogn).

The time complexity of sort part is O(nlogn).

And the traverse part is O(n)

The total time complexity is O(nlogn)

We can use an efficient sorting algorithm, like quick sort.

Hence, the algorithm is easily seem to be a polynomial time algorithm.

(c)

A: sort the jobs in a descending order of $f_i$.

Theorem: Greedy schedule A is optimal.

Pf. we define the inversion in schedule is a pair of jobs i and j such that: $J[i].f_i < J[j].f_j$ but i scheduled before j. And we assume OPT to be an optimal schedule.

Absolutely, OPT has no idle time to ensure the earliest completion time.

- If OPT has no inversions, then A = OPT
- If OPT has inversions, let (i, j) be the inversion pair. Because $\sum_{k=1}^{j} p_k + f_i < \sum_{k=1}^{j} p_k + f_j$, $\sum_{k=1}^{i} p_k + f_j < \sum_{k=1}^{j} p_k + f_j$, if we swap i and j, the former bigger completion $\sum_{k=1}^{j} p_k + f_j$ will be replaced by $O = \max(\sum_{k=1}^{j} p_k + f_i, \sum_{k=1}^{i} p_k + f_j)$. And $O < \sum_{k=1}^{j} p_k + f_j$.

Through swapping i and j, we won't increase the maximum completion time. If we strictly decrease the number of inversions. we will finally transform OPT to no inversions (the schedule A)

- Therefore A is optimal.

# Chapter 4 – Trees on Graphs

(a) $S=\{A\}$

A: $V-S=\{B,C,D,E,F\}$

| B | C | D | E | F |
|---|---|---|---|---|
| 5 | 6 | 4 | ∞ | ∞ |

Use A→D to shorten other distances.

A: $S=\{A,D\}$ $V-S=\{B,C,E,F\}$

| B | C | D | E | F |
|---|---|---|---|---|
| 5 | 6 | 4 | ∞ | 8 |

Use A→B to shorten other distances.

A: $S=\{A,D,B\}$ $V-S=\{C,E,F\}$

| B | C | D | E | F |
|---|---|---|---|---|
| 5 | 6 | 4 | ∞ | 8 |

Use A→C to shorten other distances.

A: $S=\{A,D,B,C\}$ $V-S=\{E,F\}$

| B | C | D | E | F |
|---|---|---|---|---|
| 5 | 6 | 4 | 11 | 8 |

Use A→E to shorten other distances

A: $S=\{A,B,D,C,E\}$ $V-S=\{F\}$

| B | C | D | E | F |
|---|---|---|---|---|
| 5 | 6 | 4 | 11 | 8 |

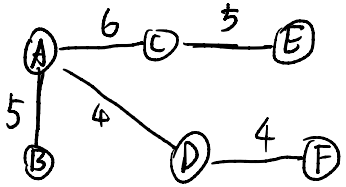
Use A→F to shorten other distances

A: $S=\{A,B,D,C,F,E\}$ $V-S=\{\}$

| B | C | D | E | F |
|---|---|---|---|---|
| 5 | 6 | 4 | 11 | 8 |

The Result for node A to all other nodes is.

| | Dis | Route |
|---|---|---|
| A→B | 5 | A→B |
| A→C | 6 | A→C |
| A→D | 4 | A→D |
| A→E | 11 | A→C→E |
| A→F | 8 | A→D→F |

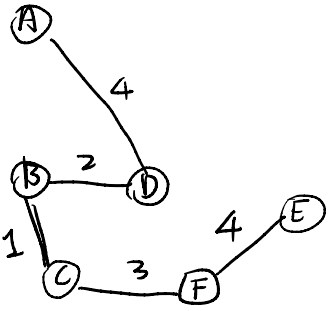


(b) Use kruskal algorithm to find the minimum spanning tree.

Follow an ascending order to sort all the edges.

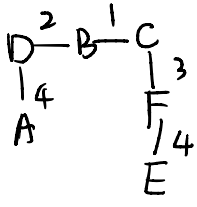
| Num | Name | Dis |
|---|---|---|
| 1 | B-C | 1 |
| 2 | B-D | 2 |
| 3 | C-D | 2 |
| 4 | C-F | 3 |
| 5 | A-D | 4 |
| 6 | D-F | 4 |
| 7 | E-F | 4 |
| 8 | A-B | 5 |
| 9 | C-E | 5 |
| 10 | A-C | 6 |

The spinning tree is

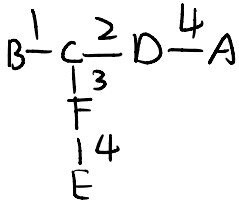



The cost of the minimum spanning tree: 4+2+1+3+4 = 14

(c) The minimum spanning tree of this graph isn't unique.




(a)




(b)

For graph(a) and graph(b), the sums of edge weights are the same while the trees in graph(a) and graph(b) have different structure.

(a)

$$d_{SPT}^{avg} = (5+6+4+11+8)\cdot\frac{1}{5}$$

$$= 6.8$$

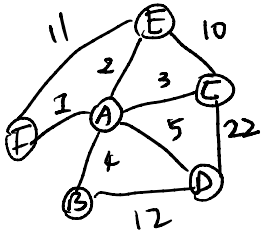
$$d_{MST}^{avg} = (4+6+7+10+14)\cdot\frac{1}{5}$$

$$= 8.2$$

Which one is greater, $d_{SPT}^{avg}$ or $d_{MST}^{avg}$ ?

$d_{MST}^{avg}$ is greater.

Does the same answer hold for any graph $G = (V, E)$ .. ?



(c)

Sometimes $d_{MST}^{avg} = d_{SPT}^{avg}$, but $d_{MST}^{avg} \geq d_{SPT}^{avg}$ is always true; the average $d_{SPT}^{avg}$ is always the smallest.

Like graph(c), the shortest path tree is the same with the minimum spinning tree.

# Chapter 4 - (Midterm, Fall 2019)

1a)

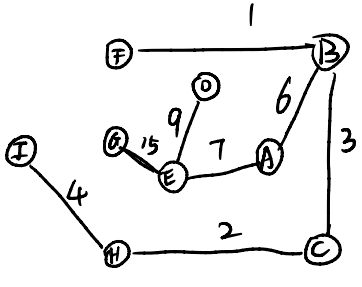
USE the kruskal Algorithm, first sort all the edges in an ascending order.

| Num | Name | Dis |
|---|---|---|
| 1 | B-F | 1 |
| 2 | H-C | 2 |
| 3 | B-C | 3 |
| 4 | I-H | 4 |
| 5 | F-I | 5 |
| 6 | A-B | 6 |
| 7 | A-E | 7 |
| 8 | E-C | 8 |
| 9 | D-E | 9 |
| 10 | F-D | 10 |
| 11 | D-B | 11 |
| 12 | D-A | 12 |
| 13 | A-C | 13 |
| 14 | E-H | 14 |
| 15 | G-E | 15 |
| 16 | D-G | 16 |
| 17 | F-G | 17 |
| 18 | I-G | 18 |
| 19 | G-H | 19 |

USE the Union-find.

| A | B | C | D | E | F | G | H | I | |
|---|---|---|---|---|---|---|---|---|---|
| A | B | C | D | E | F | G | H | I | ancestor |
| A | B | C | D | E | B | G | H | I | ancestor |
| A | B | C | D | E | B | G | H | I | |
| A | B | H | D | E | B | G | H | I | |
| A | B | B | D | E | B | G | B | I | |
| A | B | B | D | E | B | G | B | B | |
| ⋮ | | | | | | | | | |
| B | B | B | B | B | B | B | B | B. | |

② TREE




③ cost = (1+2+3+4+6+7+9+15)

= 47

(b) The MST for this graph is unique.

**Proof:** Because all the edges in G (graph presented) are distinct.

1. Assume the contrary, there are two different MSTs $A=(V,E_1)$ and $B=(V,E_2)$

2. Although A and B contain the same nodes, there is at least one edge that belongs to one but not the other. $D=\{E_1-E_2\}\cup\{E_2-E_1\}$ is the edge which meets the standard. we then assume $e_1$ is the one with least weight; the choice is unique because all the edges in D are distinct and we assume $e_1$ is in MST A.

3. Because B is MST, $\{e_1\}\cup B$ must contains a cycle C.

4. It is evident there should be an edge in C which isn't in A, let's call the edge $e_2$.

5. we previously choose the lowest-weight edge in D, what's more, $e_1$ and $e_2$ are all in D.

6. According the "cycle property" —— Let C be any cycle, and let f be the max cost edge belonging to C. Then the MST doesn't contain f.

So replacing $e_2$ with $e_1$ in MSTB will make the total cost of MSTB lower.

7. This conclusion contradicts the assumption that B is a MST. Hence, MSTA equals MSTB and the MST for this graph (with distinct edges) is unique.

It's true that the resulting H is always connected. Let $G=(V,E)$ be a connected graph, and $T=(V,E_1)$ and $T'=(V,E_2)$.

The conclusion is the k is the different edges' number, then there will be a path of length k from T to T' in H. So any two distinct spinning trees will have at least one edge different, those two spinning trees will have a path, so H is connected.

Theorem: If spinning tree $T=(V,E_1)$ and spinning tree $T'=(V,E_2)$ have k different edges, there will be a path of length k from T to T' in H.

For case: $k=1$. We assume T' has exactly one edge not in T. Then they are neighbors. From the definition of how H is constructed, we can know one edge is connected between the two nodes in H.

For case $k=n$, we assume this is true.

For case $k=n+1$, if $T'=(V,E_2)$ and $T=(V,E_1)$ have $n+1$ edges different, that is to say $|E_1-E_2|=n+1$.

we randomly choose an edge e that is in $E_2$ but is not in $E_1$. Then $T \cup \{e\}$ will have a cycle (Because adding an edge to a MST will generate a cycle). $T'$ will absolutely have $e'$ which is included in $E_1$ but isn't included in $E$.

$T'' = T \cup \{e\} - \{e'\} = (V, E_3)$, $T''$ is also a spinning tree. $|E_2 - E_3| = n$

We have hypothesized that there is a path of length n from $T'$ to $T''$ in H. Because $|E_1 - E_3| = 1$, there is a path of length 1 from T to $T''$. So there is a path of length n+1 from T to $T'$.

In conclusion, every node on graph H is reachable to other nodes. Hence, graph H is always connected.