2015 Algorithm HW4 Solutions

指導教授：謝孫源 教授

助教: 葉承翰、詹博丞、林帝丞

Question 1(10pts)

解答:

設 G=(V, E)第二小的邊 e_2 不在任一個 MST 中。

任挑一個 G 的 MST: T，則將 e_2 加入 T 中必形成一個 cycle C。

因為 C 的長度至少為 3，因此必有一邊 e 之 weight 比 e_2 大。

則將 e 自 C 中移除後，形成之 Spanning Tree 為 T'，則 wt(T')<=wt(T)。此為矛盾，

因此，第二小的邊必在 MST 中。

Question 2(10pts)

解答:

- 關鍵:使用**DFS**
- $Time\ Complexity = O(V + E)$
- $\because |E| \leq |V| - 1 \quad \therefore Time\ Complexity = O(V)$

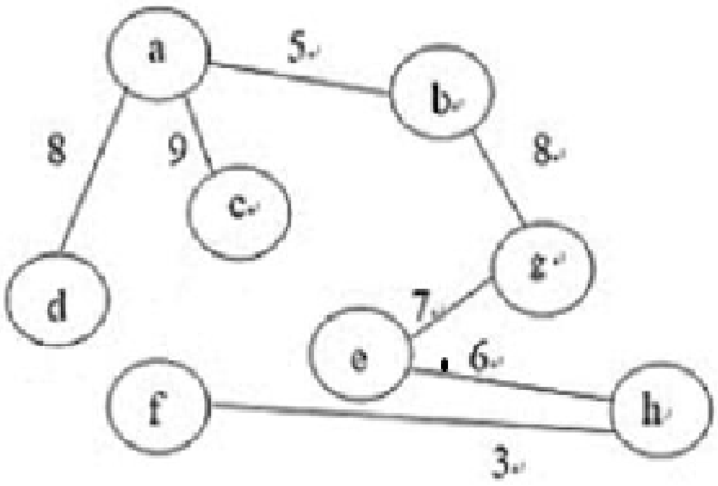
配分(10%)

- 演算法DFS 6%
- 解釋時間複雜度 4%

Question 3(a)(5pts)

解答:

- Cost is 46， kruskal’s algorithm or prim’s algorithm



Question 3(b)(5pts)

解答:

- KRUSKAL(V, E, w)

```
A ← Ø
for each vertex v ∈ V[G]
        do MAKE-SET(v)
sort E into nondecreasing order by weight w
for each (u, v) taken from the sorted list
        do if FIND-SET(u) ≠ FIND-SET(v)
              then A ← A ∪ {(u, v)}
                   UNION(u,v)
return A
```

Question 3(b)(5pts)

解答:

```
PRIM(V, E, w, r)
    Q ← Ø
    for each u ∈ V[G]
        do key[u] ← ∞
            π[u] ← NIL
            INSERT(Q, u)
    DECREASE-KEY(Q, r, 0)
    while Q ≠ Ø
        do u ← EXTRACT-MIN(Q)
            for each v ∈ Adj[u]
                do if v ∈ Q and w(u, v) < key[v]
                    then π[v] ← u
                        DECREASE-KEY(Q, v, w(u, v) )
```

Question 4(10pts)

解答:

Let T be a minimum weight spanning tree in graph G and T does not contain edge e = (u, v). We add edge e to the spanning tree T .

By the property of trees, T now contains a cycle and e is one of edges in this cycle. Now we remove from T an arbitrary edge e′ 6= e which belongs to the cycle. We obtain a new spanning tree T′.

The weight of spanning tree T′ is not more than the weight of spanning tree T , as the weight of e is not more than the weight of e′. Therefore T′ is also a minimum weight spanning tree in graph G and T′ contains e.

Question 5(10pts)

解答:

An undirected graph is acyclic (i.e., a forest) if and only if a DFS yields no back edges.

• If there is a back edge, there is a cycle.

• If there is no back edge, then by Theorem 22.10, there are only tree edges.

Hence, the graph is acyclic.

Thus, we can run DFS: if we find a back edge, there is a cycle.

Time: O(V).

(We can simply DFS.If find a back edge.there is a cycle. The complexity is O(V) instead of O(V+E).Since if there is a back edge.it must be found before seeing |V| distinct edges.This is because in a acyclic(undirected) forest, |E|<=|V|+1)

演算法:6%

時間複雜度:4%

Question 6(10pts)

解答:

Ans: True.

We can use Depth First Traversal to compute the finish times and then return the nodes in order of decreasing finishing times. We can also easily check for cycles as we do this and report no sort is possible if a cycle exists.

Question 7(10pts)

解答:

Let n be the total number of activites, $a_1, a_2, ..., a_n$.

```
GREEDY-ACTIVITY-SELECTOR-JMC(s,f)
    n = s.length
    A = {a_n}

    for m=n-1  to  1
        if f[m]<=s[k]                          //greedy step
            A={a_m} U A
            k=m

    return A
```

where n is the number of activities,
s is an n array and s[k] contains the starting time of a_k,
Assume s is monotonically increasing sorted array,
f is an n array and f[k] contains the finish time of a_k,

Question 7(10pts)

解答:

This algorithm iterates through the activities starting from the activity with the latest Starting time. If the current activity has not finished before the last activity has started, then that activity is skipped and not added to optimal solution. However, if the candidate activity, ak, does finish before the last one starts, then that activity is added to the solution. This is the greedy step and we know that the first activity with the latest starting time is going to be chosen before all the other ones because the array of activities is sorted in increasing order. In order for this approach to yield an optimal solution, it is sufficient to prove that any activity with the latest starting time belongs to a maximum-size subset of mutually compatible activities of Sk. **Claim** : Consider any nonempty subproblem Sk and let am be an activity in Sk with the last starting time. Then am is included in some maximum-size subset of mutually compatible activities of Sk.

Question 7(10pts)

解答:

Proof: Let a_i be an activity with starting time s_i and final time f_i. Let $\{a_1, a_2, ..., a_n\}$ be a set of activities monotonically increasing based on their starting time. That is, $s_1 \leq s_2 \leq s_3 \leq \ldots \leq s_n$. Let A_k be a maximum-size subset of mutually compatible activities S_k, and let a_j be the activity in A_k with the latest starting time.

Case 1: $a_j = a_m$

Then, since $a_j \in A_k$, a_j is in some maximum-size subset of mutually compatible activities of S_k

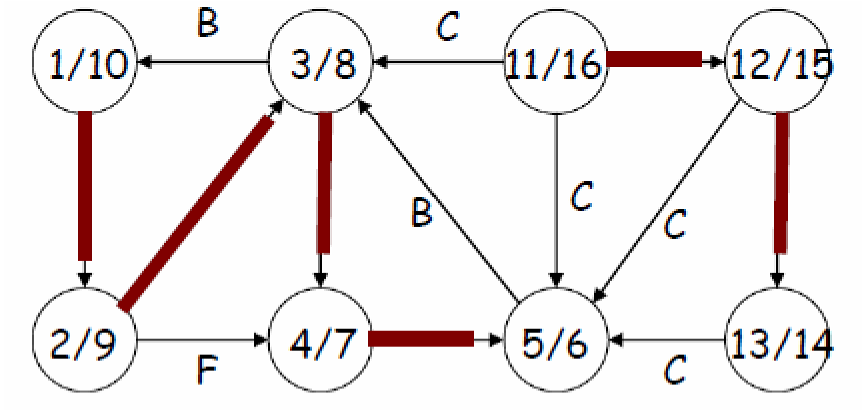
Case 2: $a_j \neq a_m$

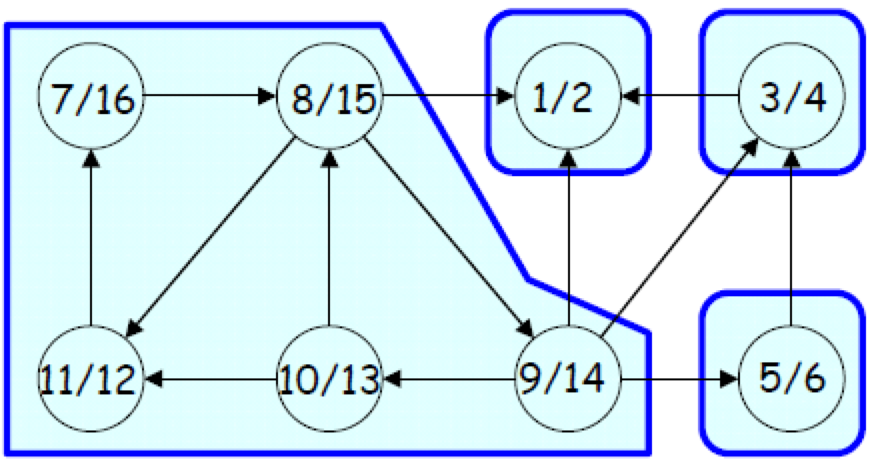
Then set $A'_k = A_k - \{a_m\} + \{a_j\}$. Since A_k is some maximum-size subset of mutually compatible activities in S_k, then $f_1 \leq f_2 \leq f_3 \leq ... \leq s_m$. Since a_j and a_m are both activities with the latest starting time in S_k, $s_m = s_j$.Then we have that $f_1 \leq f_2 \leq f_3, ... \leq s_m = s_j$ and $|A'_k| = |A_k|$. Necessarily, A'_k must be a maximum-size subset of mutually compatible activities. Since $a_j \in A'_k$, a_j is in some maximum size subset of mutually compatible activities of S_k

- 方法:5%，解釋證明: 5%

Question 8(10pts)

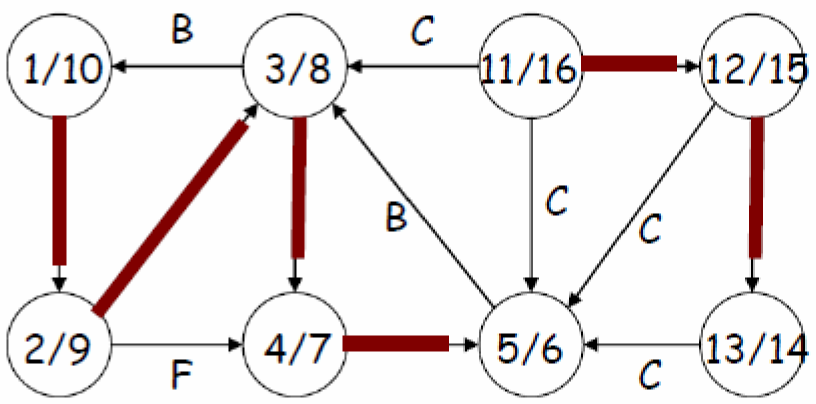
解答:





Question 9(10pts)

解答:



No, a directed graph is not acyclic because it has "back" edges.

Question 10(10pts)

解答: