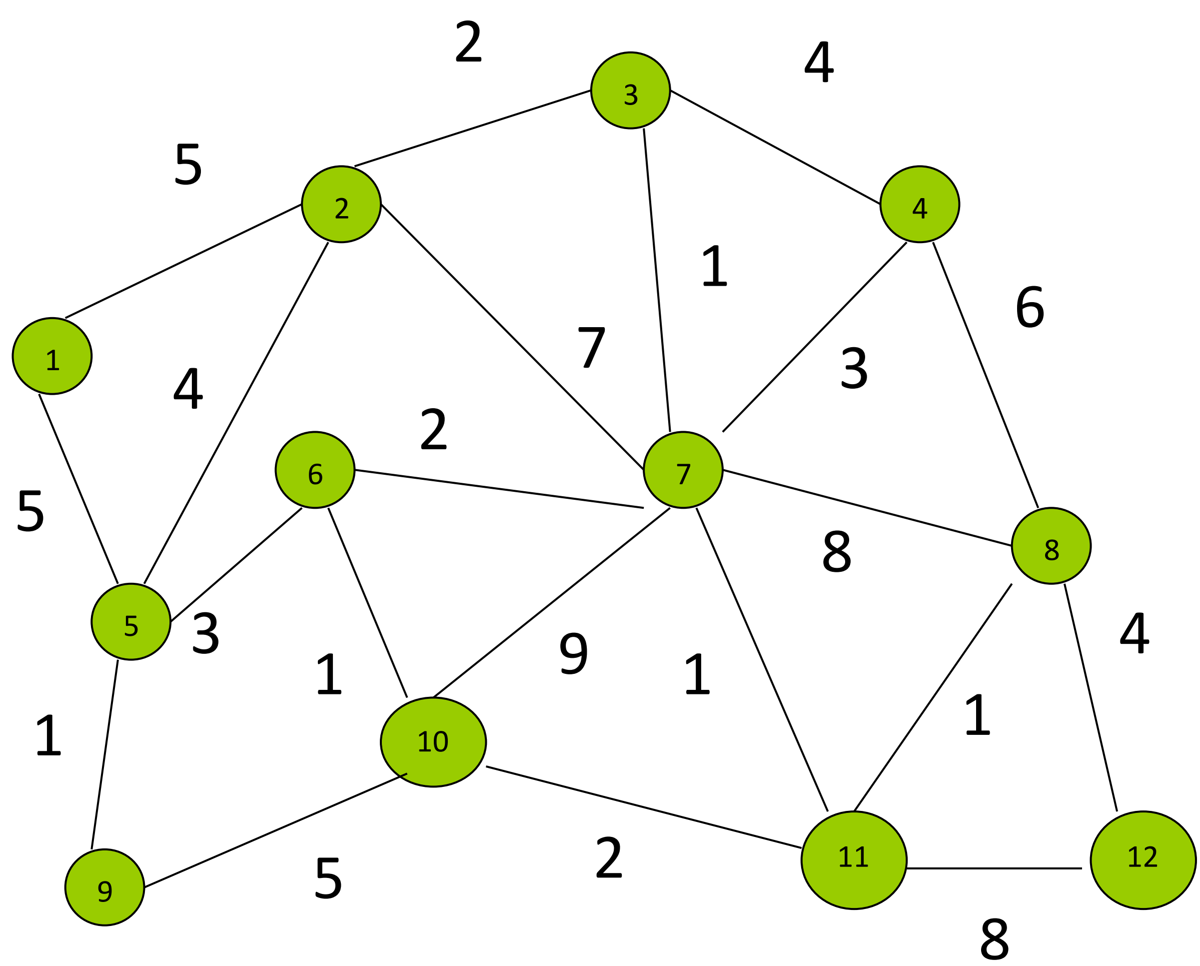
Assignment 1 Solutions

Question 1: (40 points) For the interval scheduling problem, given jobs (s, f) :
(0, 3), (2, 4), (3, 8), (4, 9), (8, 10), (4, 6), (6, 8), (9, 12), find a maximum subset of mutually compatible jobs.

Solution 1

1. Sort the jobs in increasing order of finish time
   (0, 3), (2, 4), (4, 6), (3, 8), (6, 8), (4, 9), (8, 10), (9, 12)
2. Using greedy algorithm
   Initial value: A=Φ
   (1) job (0,3), job (0,3) is compatible with jobs in A, so A = {(0,3)}
   (2) job (2,4), incompatible with job (0,3)
   (3) job (4,6), compatible, A={(0,3), (4,6)}
   (4) job (3,8), incompatible with job (4,6)
   (5) job (6,8), compatible, A={(0,3), (4,6), (6,8)}
   (6) job (4,9), incompatible with (6,8)
   (7) job (8,10), compatible, A={(0,3), (4,6), (6,8), (8,10)}
   (8) job (9,12), incompatible with (8,10)
   The job set is {(0,3), (4,6), (6,8), (8,10)}, the maximum number of compatible jobs is 4.

Question 2: (40 points) Consider the following graph.



Use Kruskal's algorithm to compute a minimum spanning tree.

Solution 2

Step 1. Sort all edges in non-decreasing order by their weights:
(8,11,1), (5,9,1), (6,10,1), (3,7,1), (7,11,1), (2,3,2), (6,7,2), (10,11,2), (5,6,3), (4,7,3), (2,5,4), (3,4,4), (8,12,4), (1,2,5), (1,5,5), (4,8,6), (2,7,7) (7,8,8), (11,12,8), (7,10,9).

Step 2. Select (8,11,1), (5,9,1), (6,10,1), (3,7,1), (7,11,1).
A = {(8,11,1), (5,9,1), (6,10,1), (3,7,1), (7,11,1)}.

Step 3. Select (2,3,2), (6,7,2) or Select (2,3,2), (10,11,2).
A = {(8,11,1), (5,9,1), (6,10,1), (3,7,1), (7,11,1) (2,3,2), (6,7,2)}.

Step 4. Select (5,6,3), (4,7,3).
A = {(8,11,1), (5,9,1), (6,10,1), (3,7,1), (7,11,1) (2,3,2), (6,7,2) (5,6,3), (4,7,3)}.

Step 5. Select (8,12,4).
A = {(8,11,1), (5,9,1), (6,10,1), (3,7,1), (7,11,1) (2,3,2), (6,7,2) (5,6,3), (4,7,3), (8,12,4)}.

Step 6. Select (1,2,5) or Select (1,5,5).
A = {(8,11,1), (5,9,1), (6,10,1), (3,7,1), (7,11,1) (2,3,2), (6,7,2) (5,6,3), (4,7,3), (8,12,4) (1,2,5)}.

Question 3. (20 points) We have a supercomputer and n PC's (of the same type) to complete the n jobs $J_1, J_2, \ldots, J_n$. Each job J_i must be processed by the supercomputer *first* with running time p_i and then it needs to be finished on one of the PC's with running time f_i. The supercomputer can only process a single job at a time. The finishing of the jobs can be performed fully in parallel on the n PC's (assuming the speed of all the PCs is the same). As soon as the i-th job in order is done on the supercomputer, it can be handed off to a PC for finishing; at that point in time the ($i+1$)-th job can be fed to the supercomputer; when the ($i+1$)-th job is done on the supercomputer, it can proceed to another PC immediately.

A schedule is an ordering of the n jobs for the supercomputer. The completion time is the earliest time at which all jobs will have finished processing on the PC's.

Give a greedy algorithm that finds a schedule with the shortest completion time.
Prove that the algorithm is correct.

Hint: You may consider the cases, where n=2 and 3, before designing an algorithm for the general case.

Solution 3

1) Schedule the jobs in the deceasing order of their running time f.

2) Proof of Correctness:

a) Let $J_1 J_2 \ldots J_n$ be a schedule. Then:
The finish time of J_1: $p_1 + f_1$

The finish time of J_2: $p_1 + p_2 + f_2$

The finish time of J_3: $p_1 + p_2 + p_3 + f_3$

…

The finish time of J_n: $p_1 + p_2 + \cdots + p_n + f_n$

The completion time = the longest among the above n finish times.

Theorem: If $J_1 J_2 \ldots J_n$ is optimal, there is a pair of neighboring jobs $J_k J_{k+1}$ in this schedule such that $f_k < f_{k+1}$, $J_1 J_2 \ldots J_{k-1} J_{k+1} J_k J_{k+2} \ldots J_n$ is optimal as well.

Proof of Theorem:

In $J_1 J_2 \ldots J_n$:

The finish time of $J_k = p_1 + \cdots + p_k + f_k$

The finish time of $J_{k+1} = p_1 + \cdots + p_k + p_{k+1} + f_{k+1}$

In $J_1 J_2 \ldots J_{k-1} J_{k+1} J_k J_{k+2} \ldots J_n$:

The finish times of all the jobs except J_k and J_{k+1} are the same as in $J_1 J_2 \ldots J_n$.

The finish time of $J_k = p_1 + \cdots + p_k + f_{k+1}$

The finish time of $J_{k+1} = p_1 + \cdots + p_k + p_{k+1} + f_k$

Noting that:

Max{ $p_1 + \cdots + p_k + f_{k+1}$, $p_1 + \cdots + p_k + p_{k+1} + f_k$ }<= Max{ $p_1 + \cdots + p_k + f_k$, $p_1 + \cdots + p_k + p_{k+1} + f_{k+1}$ }

$$\begin{array}{|c|c|}\hline p_1 + p_2 + \cdots + p_k & f_k \\ \hline \end{array} \qquad \qquad \begin{array}{|c|c|}\hline p_1 + p_2 + \cdots + p_k & f_{k+1} \\ \hline \end{array}$$

$$\begin{array}{|c|c|}\hline p_1 + p_2 + \cdots + p_k + p_{k+1} & f_{k+1} \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|}\hline p_1 + p_2 + \cdots + p_k + p_{k+1} & f_k \\ \hline \end{array}$$

Thus, the completion time of $J_1 J_2 \ldots J_{k-1} J_{k+1} J_k J_{k+2} \ldots J_n$ <= the completion time of $J_1 J_2 \ldots J_n$.

Because $J_1 J_2 \ldots J_n$ is optimal,

the completion time of $J_1 J_2 \ldots J_{k-1} J_{k+1} J_k J_{k+2} \ldots J_n$ =the completion time of $J_1 J_2 \ldots J_n$.

In other word, $J_1 J_2 \ldots J_{k-1} J_{k+1} J_k J_{k+2} \ldots J_n$ is optimal.

Based on this theorem, if $J_1 \ldots J_n$ is an optimal solution, we can exchange the orders of some job pairs step by step without destroying optimality to make it in the decreasing order. So the greedy solution is optimal.