Problem Set 7

Ekkapot Charoenwanit
Efficient Algorithms

Problem 7.1. Greedy Algorithms

The Averaged Minimum Weighted Completion Time Problem can be defined as follows.

Given a set of jobs $S = \{j_1, j_2, ..., j_n\}$, where j_i demands l_i units of CPU time and has weight w_i, find a schedule that minimizes the averaged minimum completion time of the given set of jobs.

The completion time c_i of a job j_i is the time at which j_i completes its execution.
Therefore, the averaged weighted completion time C_{avg} is $\frac{1}{n}\sum_{i=1}^{n} w_i \cdot c_i$.

For example, consider a set of three jobs $S = \{j_1, j_2, j_3\}$, where j_1 requires 1 minutes with a weight of 2, j_2 requires 2.5 minutes with a weight of 2, and j_3 requires 4 with a weight of 1.

If we schedule j_1 followed by j_3 and followed by j_2,

- the completion time of j_1 is 1 and the weighted completion time of j_1 is $2 \cdot 1 = 2$.
- the completion time of j_3 is $1 + 4 = 5$ and the weighted completion time of j_3 is $1 \cdot 5 = 5$.
- the completion time of j_2 is $1 + 5 + 2.5 = 8.5$ and the weighted completion time of j_2 is $2 \cdot 8.5 = 17$.

Therefore, the average weighted completion time of this schedule is $\frac{2+5+17}{3} = 11.3$.

1) Suppose that all the jobs have equal weights. You may use $w_i = 1$ for all $i = 1, 2, .., n$. Show that there is some optimal schedule in which jobs are picked in the increasing order of CPU time.

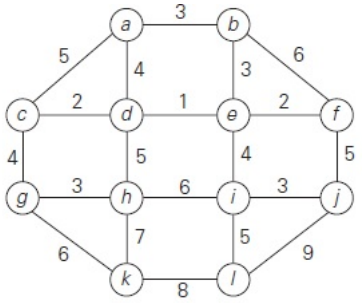
2) Suppose all the jobs demand the same amount of CPU time. Show that there is some optimal schedule in which jobs are picked in the decreasing order of weight.

3) Show that there is some optimal schedule in which jobs are picked in the decreasing order of $\frac{w}{l}$ ratio.

4) Use a cut-and-paste argument to show that there exists a unique optimal schedule for a set of n jobs, where all jobs j_i have distinct $\frac{w_i}{l_i}$ ratios.

Problem 7.2. Minimum Spanning Tree

1) Apply Prim's algorithm to the following graph.



State the vertex cut $(S, V - S)$ the algorithm uses in each step.

2) Show that the Maximum Spanning Tree Problem also exhibits a greedy-choice property.

3) How can you compute a maximum spanning tree for a connected graph using Prim's algorithm? What is the additional overhead incurred with this approach? Does this overhead cause the asymptotic behavior to differ from that of Prim's algorithm?

4) Find a maximum spanning tree of the graph in 1) using the algorithm you proposed in 3).

5) Given a connected **unweighted** graph $G = (V, E)$, how can you find a spanning tree for G using Prim's algorithm?