CS515 - Algorithms & Data Structures

Practice Assignment 3

Vy Bui - 934370552

Instructor: Professor Glencora Borradaile

The School of Electrical Engineering and Computer Science
Oregon State University

Problem 1
Job Scheduling

(a) Let $T = \{t_1, t_2, ..., t_n\}$ be the time the jobs $j_1, j_2, ..., j_n$ take.

Algorithm 1 $A(T)$

$sortedT \leftarrow sort(T)$
$lastJobCompleteTime \leftarrow 0$
$totalTime \leftarrow 0$
for t in T **do**
  $totalTime \leftarrow totalTime + lastJobCompleteTime + t$
  $lastJobCompleteTime \leftarrow lastJobCompleteTime + t$
end for
  return $sortT, totalTime$

(b) We have

$$\sum_{i=1}^{n} C_i = t_1 + (t_1 + t_2) + (t_1 + t_2 + t_3) + ... + t_n = nt_1 + (n-1)t_2 + ... + (n+1-i)t_i + (n-i)t_{i+1} + ... + t_n$$

Theorem 1: The total cost is minimum when $t_i \leq t_{i+1}$ for all i
Proof: assume that there exists some optimal job ordering that has $t_i > t_{i+1}$. Observe that there are (n + 1 - i) of t_i terms and (n - i) of t_{i+1} terms in the above summation. If we swap t_i and t_{i+1}, the summation will have one more t_{i+1} term and one less t_i term. Because $t_i > t_{i+1}$, the total cost after the swap will reduce, thus producing a not worse solution. From some optimal solution O, we can swap these inversions ($t_i > t_{i+1}$) until there is no inversions left in the ordering, which is exactly the solution of our greedy algorithm. And each swap guarantees to produce at least equally good result.

(c)
The algorithm takes $O(nlogn)$ to sort the list of jobs by time needed to complete the job. It then takes $O(n)$ time to iterate through the sorted list and accumulate the total time. The ordering is the order of the sorted list. In total, it takes $O(nlogn)$ time.

Problem 2
A wrong greedy algorithm for the Knapsack problem

(a)

(b)

(c)

Problem 3
A randomized algorithm for generating biased random bits

- mutually independent - p(F) = p(T) = 0.5

(a)

(b)

(c)

Problem 4
Tax Screening System