DDA 6050: Homework #3

Due on Nov 21, 2021

Professor Yixiang Fang

Haoyu Kang

Problem 1

First we assume that the commission is always zero. Let k denote a currency which appears in an optimal sequence ss of trades to go from currency 11 to currency nn. p_k denote the first part of this sequence which changes currencies from 1 to k and q_k denote the rest of the sequence. Then p_k and q_k are both optimal sequences for changing from 1 to k and k to n respectively. To see this, suppose that p_k wasn't optimal but that p'_k was. Then by changing currencies according to the sequence $p'_k q_k$ we would have a sequence of changes which is better than ss, a contradiction since ss was optimal. The same argument applies to q_k

Now suppose that the commissions can take on arbitrary values. Suppose we have currencies 1 through 6, and $r_{12} = r_{23} = r_{34} = r_{45} = 2, r_{13} = r_{35} = 6$ and all other exchanges are such that $r_{ij} = 100$ $Letc_1 = 0, c_2 = 1$ and $c_k = 10$ for $k \geq 3$

The optimal solution in this setup is to change 1 to 3, then 3 to 5, for a total cost of 13. An optimal solution for changing 1 to 3 involves changing 1 to 2 then 2 to 3, for a cost of 5, and an optimal solution for changing 3 to 5 is to change 3 to 4 then 4 to 5, for a total cost of 5. However, combining these optimal solutions to subproblems means making more exchanges overall, and the total cost of combining them is 18, which is not optimal.

Problem 2

We provide a pseudocode which grasps main ideas of an algorithm.

```
1   IS-INDEPENDENT(A)
2   n = A.length
3   let Nts[0..n] be an array filled with 0s
4   for each a in A
5       if a.deadline >= n
6           Nts[n] = Nts[n] + 1
7       else
8           Nts[d] = Nts[d] + 1
9   for i = 1 to n
10      Nts[i] = Nts[i] + Nts[i - 1]
11  // at this moment, Nts[i] holds value of N_i(A)
12  for i = 1 to n
13      if Nts[i] > i
14          return false
15  return true
```

Problem 3

$$\pi = \{0, 0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2, 3, 4, 5, 6, 7, 8\} \tag{1}$$

Problem 4

Problem 4 continued on next page...

```
DYNAMIC–ACTIVITY–SELECTOR(s, f, n)
let c[0..n + 1, 0..n + 1] and act[0..n + 1, 0..n + 1] be new tables
for i = 0 to n
    c[i, i] = 0
    c[i, i + 1] = 0
c[n + 1, n + 1] = 0
for l = 2 to n + 1
    for i = 0 to n − l + 1
        j = i + l
        c[i, j] = 0
        k = j − 1
        while f[i] < f[k]
            if f[i]  s[k] and f[k]  s[j] and c[i, k] + c[k, j] + 1 > c[i, j]
                c[i, j] = c[i, k] + c[k, j] + 1
                act[i, j] = k
            k = k − 1
print "A maximum size set of mutually compatible activities has size" c[0, n + 1]
print "The set contains"
PRINT–ACTIVITIES(c, act, 0, n + 1)

PRINT–ACTIVITIES(c, act, i, j)
    if c[i, j] > 0
        k = act[i, j]
        print k
        PRINT–ACTIVITIES(c, act, i, k)
        PRINT–ACTIVITIES(c, act, k, j)
```

GREEDY-ACTIVITY-SELECTOR runs in $\Theta(n)$time and
DYNAMIC-ACTIVITY-SELECTOR runs in $O(n^3)$

Problem 5

Instead of grouping together the two with lowest frequency into pairs that have the smallest total frequency, we will group together the three with lowest frequency in order to have a final result that is a ternary tree. The analysis of optimality is almost identical to the binary case. We are placing the symbols of lowest frequency lower down in the final tree and so they will have longer codewords than the more frequently occurring symbols.

Problem 6

Decision version:
given n, m and t1,t2, . . . ,tn and a target T, is there a schedule with makespan at most T?
Problem is NP-Complete:
reduce from Subset Sum.
greedy algorithm(2-approximation algorithm):
1. Consider the jobs in some fixed order
2. Assign job j to the machine with lowest load so far

Problem 6 continued on next page...

for *each machine i* **do**
  $T_i = 0$ (* initially no load *)
  $A(i) = \emptyset$ (* initially no jobs *)
for *each job j* **do**
  Let i be machine with smallest load
  $A(i) = A(i) \cup j$ (* schedule j on i *)
  $T_i = T_i + t_j$ (* compute new load *)

Modified Greedy(3/2-approximation algorithm):
Sort the jobs in descending order of processing time, and process jobs using greedy algorithm

for *each machine i* **do**
  $T_i = 0$ (* initially no load *)
  $A(i) = \emptyset$ (* initially no jobs *)
for *each job j in descending order of processing time* **do**
  Let i be machine with smallest load
  $A(i) = A(i) \cup j$ (* schedule j on i *)
  $T_i = T_i + t_j$ (* compute new load *)

Problem 7

To show a problem is NP-Complete, then proof that the problem is in NP and any NP-Complete problem.
3-coloring problem is in NP:
This can be done in the following way:For each edge u, v in graph G verify that the color c(u) != c(v)
Hence, the assignment can be checked for correctness in the polynomial-time of the graph with respect to its edges O(V+E).
3-coloring problem is NP-Hard:
In order to prove that the 3-coloring problem is NP-Hard, perform a reduction from a known NP-Hard problem to this problem. Carry out a reduction from which the 3-SAT problem can be reduced to the 3-coloring problem.Let us assume that the 3-SAT problem has a 3-SAT formula of m clauses on n variables denoted by x1, x2, …, xn. The graph can then be constructed from the formula in the following way:
1. For every variable xi Construct a vertex vi In the graph and a vertex vi' denoting the negation of the variable xi.
2. For each clause c in m, add 5 vertices corresponding to values c1, c1, …, c5.
3. Three vertices of different colors are additionally added to denote the values True, False, and Base (T, F, B) respectively.
4. Edges are added among these three additional vertices T, F, B to form a triangle.
5. Edges are added among the vertices vi and vi' and Base (B) to form a triangle.

Problem 8

Suppose for the original tree T = <V, E>. That is, the vertex set is V, the edge set is E. Also suppose the cover set = C. The algorithm can be described as follows:

Initialize the cover set C =∅
while V =∅ **do**
  Identify a leaf vertex v
  Locate u = parent(v), the parent vertex of v.
  C = $C \cup u$.
  Remote all the edges incident to u
return C

Problem 9

Find T' in TT

Problem 10

Showing the problem is in NP:
Let $C = \{S_1, S_2, \ldots, S_n\}$ be the certificate.We assume that each $S \in C$, uniquely covers at least one $x \in X$, so:

$$\begin{aligned} |C| &\leq |X| \\ \forall S \in C : |S| &\leq |X| \end{aligned} \tag{2}$$

This gives us that we can verify that $n \leq k$ and that C covers X in $O(|x|^2)$,which proves the problem is in NP
Showing the problem is NP-hard
We will show that the decision version of the set-covering problem is NP-hard by showing:

$$Vertex - Cover \leq_p Set - Cover$$