Student Name: Duc Tu Luong
Last 3-digit ID: 122

Homework #6

1. a. Problem model:
   A company wants to ship their PC equipment supply with the value of s_i (in pound) of week i. They can choose either freight company A or B. Company A charges a fixed rate r per pound (it costs r * s_i to ship a week's supply). Company B charges a fixed amount c per week, no matter how many pounds per week, however, company B always makes contract in block of 4 consecutive weeks at a time. Give a polynomial-time algorithm that takes a sequence of supply values s_1, s_2… s_n a return a schedule of minimum cost.

   b. This problem belongs to class P

   c. Polynomial-time algorithm:
   - The overall idea is using dynamic programming to choose between company A or company B for each week, and optimal solution up to this week, considering to company B's constraint. Let say for week i, we can choose between:
   Optimal price if choosing company A: r * s_i + optimal(i - 1) = A
   Optimal price if choosing company B: 4 * c + optimal(i - 4) = B
   We return min(A, B) for each week.
   - Pseudo code:

```
Optimal (n)
    If (n = 0)
        return 0
    ElseIf (n < 4)
        return r * sn
    Else
        A = r * sn + Optimal (i - 1)
        B = 4 * c + Optimal (i - 4)
        return min(A, B)
    EndIf
End
```

   - Example: suppose r = 1, c = 10, and the sequence of values is:
   11, 9, 9, 12, 12, 12, 12, 9, 9, 11

   Week 1: optimal solution is 11 (select company A)
   Week 2: optimal solution is 20 (select company A)
   Week 3: optimal solution is 29 (select company A)
   Week 4: optimal solution is 40 (select company B)
   Week 5: optimal solution is 51 (select company B)
   Week 6: optimal solution is 60 (select company B)

Week 7: optimal solution is 69 (select company B)
Week 8: optimal solution is 78 (select company A)
Week 9: optimal solution is 87 (select company A)
Week 10: optimal solution is 98 (select company A)
=> the optimal price is 98, the sequence of freight company is A, A, A, B, B, B, B, A, A, A
- Time complexity is O(n)

2. a. Problem model:
   We have n processes running on a system that can run concurrently, but certain pairs of jobs cannot. In a next k time steps of the system, is it possible to schedule each process to run at least once?

   b. This problem belongs to NP-Complete

   c. Solution:
   - We model this problem by an undirected graph G = (V, E). We construct a graph with vertices representing n jobs in the system. For pairs of jobs which cannot run concurrently, we create an edge between these nodes.
   - We can use k-coloring algorithm to solve this problem, all the colored nodes in any step can run concurrently without conflict. So, we need to solve k-coloring problem to solve this problem. When we have more than 3 nodes, this problem can reduce to 3-coloring problem by taking 3-node graph and adding (k - 3) nodes graph to this 3-node graph.

3. a. Problem model:
   We have 2 databases, with n number of numerical values, so there are 2n values total and assume all values are unique. Calculate databases median using as few queries as possible.

   b. This problem belongs to class P

   c. Solution:
   - We solve this problem by first comparing middle element of 2 databases. Let A and B be the names of 2 databases, let k = n/2, A(k) and B(k) be the medians of databases A and B. If A(k) < B(k), then B(k) is greater than any values from A(1) to A(k) any values from B(1) to B(k - 1). There are at least n records in A and B which are smaller or equal to max(A(k), B(k)). Thus, the median of A and B is not greater than max(A(k), B(k)), and not smaller than min(A(k), B(k)).
   Based on this observation, we can adjust k accordingly in A and B. Finally, we return the smaller values of k^{th} position among A and B.

- Pseudo code:

```
Let A and B be databases
Let kA = kB = n/2
For i=2 to logn
    vA = A(kA)
    vB = A(kB)
    If (vA > vB)
        kA = kA – n/2^i
        kB = kB + n/2^i
    Else
        kA = kA + n/2^i
        kB = kB – n/2^i
    EndIf
EndFor
return min(v1, v2)
```

- Step by step:

First we let median position of A and B is equal to n/2, then we compare 2 values of k^{th} position in A and B, if value of k^{th} position in A is greater, we discard last half of A and first half of B and update kA to median position of first half of A and kB to median position of last half of B accordingly (we take n divide 2^i each time because the size of A and B reduce half each iteration). Finally, we return the smaller of A(kA) and B(kB).

- Time complexity: O(logn)

4. a. Problem model:

   A photocopying service with a single large machine wants to order their customers' jobs every day to make their customers happy. Each customer's job i takes t_i time to finish and weights w_i, the importance of the job. Let C_i denotes the finishing time of the job i. Provide a job scheduling to minimize the weighted sum of the completion times, Sum(for i =1 to n) (w_iC_i)

   b. This problem belongs to class P

   c. Solution

   - We schedule the jobs in decreasing order of w/t. We use exchange argument method to prove this. Let say we have an optimal schedule other than this schedule which schedule job j before job i and we already know that $w_it_i > w_jt_j$. If we schedule j before i, then $W_{ji} = w_jt_j + w_i(t_j + t_i)$, then if we swap i before j, we will have $W_{ij} = w_it_i + w_j(t_i + t_j)$. Since $w_it_i > w_jt_j$, $W_{ji} – W_{ij} >= 0$. Thus, our schedule does not increase sum and may reduce sum in total, if we continue to swap for all jobs in the optimal schedule which do not follow our schedule, we will have an optimal schedule which is exactly like our schedule.

- Pseudo code:
Sort jobs in decreasing order of w/t for each job

Process jobs through large machine following this order

- Time complexity: O(nlogn), because we must sort first, sorting takes logn time.

5. a. Problem model:
Given a directed graph G = (V, E), and specified node **s** and node **t,** a number **c**, the number of users who each requests a path in G from **s** to **t**, is it possible to select at least **c** of the paths so that no two of the selected paths share any edges?

b. This problem belongs to class P

c. Solution:
- We construct a graph G = (V, E), with source s and sink t, c vertices represent c users. For each node, we create an edge from source s to this node with capacity of 1 and an edge from this node to sink t also with capacity of 1.
- Now we have a max-flow problem, we use Ford-Fulkerson algorithm to solve this problem, if the max-flow >= c, then it is possible to reserve c paths for c users, otherwise, it is impossible.
- Let maxFlow be the possible maximum flow in G, E is the number of edges in G. Time complexity for this problem is O(maxFlow * E).

6. a. Problem model:
You have n final projects which must be completed, each project has a specified utility points ranging from -20 to 30, each project also has pre-requisites projects. Select a set of projects to accomplish maximizing your total utility points from them.

b. This problem belongs to class P

c. Solution
- We construct a directed graph G = (V, E) with a source s and sink t. Vertices in G represent projects. Each project will have incoming edges from pre-requisites projects with capacity of ∞. Projects which generate negative points will have outgoing edges to sink t with capacity of that negative points. Projects which generate positive points will have incoming edges from source s to them with capacity of that positive points.
- To maximize utility points, we need to select a set of nodes (projects) which will generate maximum flow to sink t. Now this problem becomes max-flow problem in a directed graph, we solve this by implementing Ford-Fulkerson algorithm.
- Let maxFlow be the possible maximum flow in G, E is the number of edges in G. Time complexity for this problem is O(maxFlow * E).

7. a. Problem model:
   A camp is supposed to have at least one counselor who's skilled at each of the n sports covered by the camp. They have m potential counsellors, and for each of the n sports, there is some subset of the m applicants qualified. For a given number k < m, is it possible to hire at most k of the counselors and have at least one counselor qualified in each of the n sports?

   b. This problem belongs to class NP-Complete

   c. Solution:
   - We will call this problem Efficient Recruiting problem, we know that this problem is in NP class, because given a set of k counsellors, we can check in polynomial time that if at least one counsellor is qualified in each of n sports.
   - Vertex Cover problem is known to be in NP-Complete class.
   - Now we prove that Vertex Cover problem $\leq_P$ Efficient Recruiting Problem. Suppose we have an instance for Efficient Recruiting problem, and we construct a graph G = (V, E) and a number k for Vertex Cover problem. We need to reduce the Vertex Cover problem to an Efficient Recruiting problem. We prove that by assigning each counsellor to a vertex and edges represent sports. If a counsellor is qualified for a sport, there will be an edge coming out from the node that representing that counsellor.
   Then we check if there is a subset of k counsellors who are qualified for all sports.
   If there exists a subset of k counsellors who are qualified for all sports <==> graph G contains a vertex cover of size k
   => the instance of Efficient Recruiting problem returns "Yes", which means all sports are covered by at least one counsellor, and there is a subset of k counsellors who are qualified for all sports, thus, this subset of nodes is a vertex cover for graph G
   <= the Vertex Cover Problem is "Yes", which means there is a subset of nodes which is a vertex cover of graph G with size k, and these nodes represent counsellors in Efficient Recruiting problem, therefore, there exists a subset of k counsellors who are qualified in all sports.
   - We have proved that this problem belongs to NP-Complete class.