COMP 271: Final Exam
2002 Fall Semester
17 December 2002
8:30 am – 11:30 am

You have 3 hours to solve 5 problems for a total of 100 marks. Answer all problems in the space provided. You may also use the back of the test papers. Good luck!

Student name: ____________________ **Student ID:** ____________________

Problem	Your mark	Maximum
1		20
2		20
3		20
4		20
5		20

Your total marks: ____________

Problem 1. (20 points) Answer the following in 2–4 sentences. In each case provide a clear and precise explanation of your answer.

(a) (3 points) Does the problem of deciding whether there exists a cycle in an undirected graph belong to the class NP?

(b) (3 points) Does the problem of deciding whether there exists a cycle in an undirected graph belong to the class P?

(c) (3 points) Consider the following problem: Given a set of n integers, sort them in non-decreasing order. Does this problem belong to the class P?

(d) (3 points) Consider the following problem: Given an undirected graph G, decide whether or not it contain a clique of size 10. Does this problem belong to the class P?

(e) (4 points) Assume that the 3-SAT decision problem does not belong to the class P. Does it then follow that the CLIQUE decision problem does not belong to the class P?

(f) (4 points) Assume that the 3-SAT decision problem does not belong to the class P. Does it then follow that no problem in the class NP can be solved in polynomial time?

Problem 2. (20 points)

Let a_i, $i = 1, 2, \ldots, 8$ be eight symbols. You are given a file in which symbol a_i occurs f_i times (i.e. f_i is the frequency of symbol a_i) for $i = 1, 2, \ldots, 8$. If ℓ_i is the number of bits in the codeword for symbol a_i, then the total number of bits used to encode the file is

$$\sum_{i=1}^{8} f_i \ell_i.$$

(a) (4 points) Explain why the total number of bits used to encode the file is at most $3 \sum_{i=1}^{8} f_i$.

(b) (8 points) Let $f_i = i$ for $i = 1, 2, \ldots, 8$, that is a_1 has frequency $f_1 = 1$, a_2 has frequency $f_2 = 2$, etc. Construct an optimal binary prefix code tree for this set of frequencies, using Huffman's algorithm.

(c) (4 points) From your answer in part (b), determine the corresponding variable length encoding of each of the symbols, $a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8$.

(d) (4 points) Determine the total number of bits used to encode the file, for the code of part (c).

Problem 3. (20 points)

Suppose you are given three strings of characters: $X = x_1 x_2 \cdots x_n$, $Y = y_1 y_2 \cdots y_m$, $Z = z_1 z_2 \cdots z_p$, where $p = n + m$. Z is said to be a *shuffle* of X and Y iff Z can be formed by interleaving the characters from X and Y in a way that maintains the left-to-right ordering of the characters from each string. The goal of this problem is to design an efficient dynamic-programming algorithm that determines whether Z is a shuffle of X and Y.

(a) (5 points) Show that *cchocohilaptes* is a shuffle of *chocolate* and *chips*, but that *chocochilatspe* is not.

(b) (10 points) For any string $A = a_1 a_2 \cdots a_r$, let $A_i = a_1 a_2 \cdots a_i$ be the substring of A consisting of the first i characters of A. For example, if A is *chocolate*, then A_3 is *cho* and A_6 is *chocol*.

Define $f(i, j)$ to be 1 if Z_{i+j} is a shuffle of X_i and Y_j, and 0 otherwise. Derive a recursive formula for $f(i, j)$. Remember to include the basis cases. Briefly explain your derivation.

(c) (5 points) Give an efficient algorithm for determining whether Z is a shuffle of X and Y. Analyze the running time of your algorithm.

Problem 4. (20 points)

Recall that Dijkstra's algorithm is used to compute the lengths of the shortest paths in a weighted directed graph $G = (V, E)$, from a source vertex s. The goal of this problem is to establish the correctness of Dijkstra's algorithm. A proof skeleton has been given below but the explanations to many of the statements are missing. This is indicated by writing the word "WHY???" after the statements and leaving some space for you to answer the WHYs. Try to answer as many of the WHYs as you can, clearly and precisely. If you find that you cannot answer some of the WHYs, do not get stuck there, instead go to the ones that you can answer.

We will assume that the weights on all the edges are *strictly positive*. For any two vertices u and v, we let $w(u, v)$ denote the weight of the edge from u to v. Here is the pseudo-code for Dijkstra's algorithm:

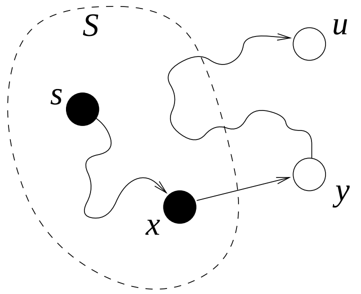
```
Dijkstra(G,w,s)
{
    for each u in V do                          // initialization
        d[u] = +infinity;
    d[s] = 0;
    Q = build initial priority queue with all vertices;
    S = empty-set;
    while (Non_Empty(Q)) do                     // until all vertices processed
    {
        u = Extract_Min(Q);                     // select vertex with smallest d[u]
        Add vertex u to set S;
        for each v in Adj[u] do
        {
            if (d[u] + w(u,v)) < d[v] then      // Relax(u,v)
            {
                d[v] = d[u] + w(u,v);
                Decrease_Key(Q, v, d[v]);
            }
        }
    }
}
```

On the next page you will establish correctness of Dijkstra's algorithm using the proof skeleton provided.

For any vertex $v \in V$, let $\delta(s, v)$ denote the length of the shortest path from s to v. In order to establish the correctness of Dijkstra's algorithm we need to prove the following lemma.

Lemma: When a vertex u is added to S, $d[u] = \delta(s, u)$.

Proof Skeleton: For the purpose of contradiction, let u be the first vertex added to S for which $d[u] \neq \delta(s, u)$. We focus our attention on the situation *just before* vertex u is added to S. Let p denote the shortest path from s to u in the graph G. Since $s \in S$ and $u \in V - S$, the path p must take an edge out of S. Starting from s, let y be the first vertex along p such that $y \in V - S$, and let x be the vertex on p just before y. See figure. Note that it may be that $x = s$.



In the first part of the proof we will show that, just before u is added to S, $d[y] = \delta(s, y)$ (in particular, this would imply that $y \neq u$). To prove this, we first observe that

$$d[x] = \delta(s, x). \tag{1}$$

WHY???

Next we claim that

$$d[y] \leq d[x] + w(x, y). \tag{2}$$

WHY??? (*Hint:* Think about what the algorithm does just after it adds x to S.)

Combining Eq. (1) and Eq. (2) we get

$$d[y] \leq \delta(s, x) + w(x, y). \tag{3}$$

Now,

$$\delta(s, x) + w(x, y) = \delta(s, y). \tag{4}$$

WHY???

Combining Eq. (3) and Eq. (4) we get

$$d[y] \leq \delta(s, y). \tag{5}$$

We also know that

$$d[y] \geq \delta(s, y). \tag{6}$$

WHY???

Combining Eq. (5) and Eq. (6) we get

$$d[y] = \delta(s, y). \tag{7}$$

It follows that $y \neq u$. Next, we claim that

$$\delta(s, y) < \delta(s, u). \tag{8}$$

WHY???

Also,

$$\delta(s, u) \leq d[u]. \tag{9}$$

WHY???

Combining Eq. (7), Eq. (8) and Eq. (9) we get

$$d[y] = \delta(s, y) < \delta(s, u) \leq d[u]. \tag{10}$$

Thus y would have been added to S before u. WHY???

This is a contradiction. WHY???

This completes the proof.

Problem 5. (20 points)

Show that the Modified Hamiltonian path problem (MHP) is NP-complete. You may assume the fact that the Hamiltonian path problem (HP) is NP-complete. Here are the problem definitions.

Hamiltonian Path (HP): Given an undirected graph $G = (V, E)$, is there a simple path that visits all the vertices of G exactly once?

Modified Hamiltonian Path (MHP): Given an undirected graph $G = (V, E)$ and two distinct vertices u and v, is there a simple path starting at u and ending at v that visits all the vertices of G exactly once?