CONCORDIA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

COMP 6651/4: Algorithm Design Techniques

Fall 2014

MidTerm - Close book exam - 3 hours

Instructor: Professor B. Jaumard
Office: EV 003.115
Email: bjaumard@cse.concordia.ca

First Name	**Last Name**	**ID#**

Question 1. **(Approximation Algorithm)** (20 points.)

Suppose you are given a set of positive integers $A = \{a_1, a_2, \ldots, a_n\}$ and a positive integer B. A subset $S \subseteq A$ is called feasible if the sum of the numbers in S does not exceed B:

$$\sum_{a \in S} a \leq B.$$

The sum of the numbers in S will be called the total sum of S.

You would like to select a feasible subset S of A whose total sum is as large as possible.

Example. If $A = \{8, 2, 4\}$ and $B = 11$, then the optimal solution is the subset $S = \{8, 2\}$.

(i) Here is an algorithm for this problem.

```
Initially $S = \emptyset$. Define $T = 0$.
For $i = 1, 2, ..., n$
    If $T + a_i \leq B$ then $S \leftarrow S \cup \{a_i\}$ ; $T \leftarrow T + a_i$
Endif
Endfor
```

Give an instance in which the total sum of the set S returned by this algorithm is less than half the total sum of some other feasible subset of A.

(ii) Give a polynomial-time approximation algorithm for this problem with the follow- ing guarantee: It returns a feasible set $S \subseteq A$ whose total sum is at least half as large as the maximum total sum of any feasible set $S' \subseteq A$.

You should give a proof that your algorithm has this property, and give a detailed analysis of its running time.

Question 2. **(Dynamic Programming)** (20 points.) (Assembly Line Scheduling)

Suppose you are running a lightweight consulting business just you, two associates, and some rented equipment. Your clients are distributed between the East Coast and the West Coast, and this leads to the following question.

Each month, you can either run your business from an office in New York (NY), or from an office in San Francisco (SF). In month i, you will incur an operating cost of N_i if you run the business out of NY; you will incur an operating cost of S_i if you run the business out of SF. (It depends on the distribution of client demands for that month.) However, if you run the business out of one city in month i, and then out of the other city in month $i+1$, then you incur a fixed moving cost of M to switch base offices. Given a sequence of n months, a plan is a sequence of n locations each one equal to either NY or SF such that the ith location indicates the city in which you will be based in the ith month. The cost of a plan is the sum of the operating costs for each of the n months, plus a moving cost of M for each time you switch cities. The plan can begin in either city.

The problem is: Given a value for the moving cost M, and sequences of operating costs $N_1, N_2, \ldots, N_n$ and $S_1, S_2, \ldots, S_n$, find a plan of minimum cost. (Such a plan will be called optimal.)

Example. Suppose $n = 4, M = 10$, and the operating costs are given by the following table.

	Month 1	Month 2	Month 3	Month 4
NY	1	3	20	30
SF	50	20	2	4

Then the plan of minimum cost would be the sequence of locations [NY,NY, SF, SF], with a total cost of 1 + 3 + 2 + 4 + 10 = 20, where the final term of 10 arises because you change locations once.

(*i*) Show that the following algorithm does not correctly solve this problem, by giving an instance on which it does not return the correct answer.

```
For $i = 1$ to $n$
    If $N_i < S_i$ then Output NY in Month $i$ Else Output SF in Month $i$
```

In your example, say what the correct answer is and also what the above algorithm finds.

(*ii*) Give an example of an instance with $n = 4$ in which every optimal plan must move (i.e., change locations) at least three times. Provide an explanation, of at most three sentences, saying why your example has this property.

(*iii*) Give an algorithm that takes values for n, M, and sequences of operating costs $N_1, N_2, \ldots, N_n$ and $S_1, S_2, \ldots, S_n$, and returns the cost of an optimal plan. The running time of your algorithm should be polynomial in n. You should prove that your algorithm works correctly, and include a detailed analysis of the running time.

Hint. First compute $\mathrm{OPT}_N(n)$ (minimum cost of a plan on months $1, 2, \ldots, n$ ending in NY) and $\mathrm{OPT}_S(n)$ (minimum cost of a plan on months $1, 2, \ldots, n$ ending in SF) and then deduce from these two expressions the cost of an optimal plan.

Question 3. (20 points)

Let S be a set of n disjoint line segments in the plane, and let p be a point not on any of the line segments of S. We wish to determine all line segments of S that p can see, that is, all line segments of S that contain some point q so that the open segment $p - q$ does not intersect any line segment of S. Give an $O(n \log n)$ time algorithm for this problem that uses a rotating half- line with its endpoint at p.

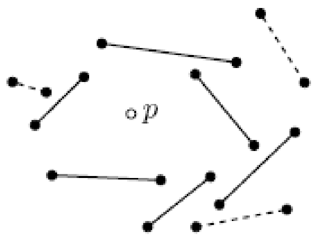

Figure 1: Visible segments from point p

Question 4. CSM (Cyclic String Matching) Problem). (20 points)

The input is two strings of characters $A = a_1 a_2 \dots a_n$ and $B = b_1 b_2 \dots b_n$. Design an $O(n)$ time algorithm to determine whether B is a cyclic shift of A. In other words, the algorithm should determine whether there exists an index k, $1 \leq k \leq n$ such that $a_i = b_{(k+i) \bmod n}$, for all i, $1 \leq i \leq n$.

Question 5. (20 points) (Coursepack 235)

(i) Describe and analyze an algorithm to compute the size of the largest connected component of black pixels in an $n \times n$ bitmap $B[1..n, 1..n]$. For example, given the bitmap below as input (see Figure 2), your algorithm should return the number 9, because the largest connected black component (marked with white dots on the right) contains nine pixels.

(ii) Design and analyze an algorithm BLACKEN(i, j) that colors the pixel $B[i, j]$ black and returns the size of the largest black component in the bitmap. For full credit, the amortized running time of your algorithm (starting with an all-white bitmap) must be as small as possible. For example, at each step in the sequence below (see Figure 3), we blacken the pixel marked with an X. The largest black component is marked with white dots; the number underneath shows the correct output of the BLACKEN algorithm.

(iii) What is the worst-case running time of your BLACKEN algorithm?

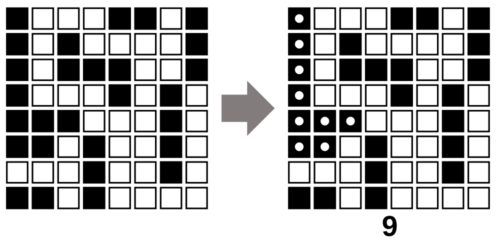

Figure 2: A bitmap

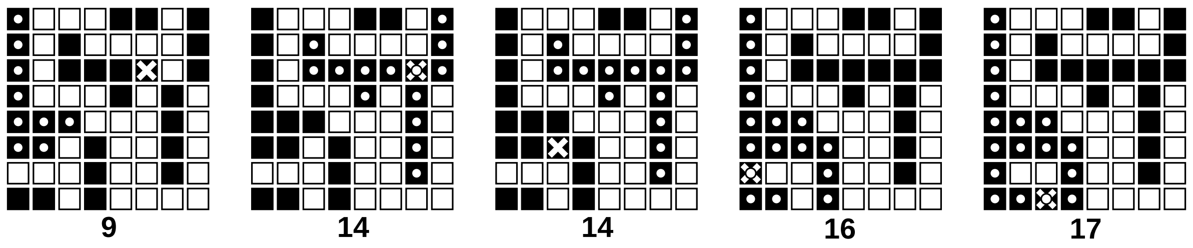

Figure 3: A Sequence