CS 341, Spring 2012
Algorithms
Timothy Chan

University of Waterloo
Final Examination

Last Name: ______________ First Name: ______________

Signature: ______________

ID number: SOLUTIONS

- Date: August 2, 2012.
- Start Time: 9:00am. End Time: 11:30am.
- Number of pages (including cover and one blank page): 15.
- No additional materials are allowed.
- Print your initials at the top of each page (in case a page gets detached).
- All answers should be placed in the spaces given. Backs of pages may be used as scratch papers and will not be marked (unless you clearly indicate otherwise). If you need more space to complete an answer, you may use the blank page at the end.
- Cheating is an academic offense. Your signature on this exam indicates that you understand and agree to the University's policies regarding cheating on exams.

Q	Marks	Init.
1	/8	
2	/30	
3	/16	
4	/11	
5	/13	
6	/22	
Total	/100	

1. [*8 marks*] *Analysis of divide-and-conquer algorithms.* Consider the following pseudocode:

```
strange(a_1, ..., a_n):
1.  if n ≤ 2012 then return
2.  strange(a_1, ..., a_⌊n/6⌋, a_⌊3n/6+1⌋, ..., a_⌊4n/6⌋)
3.  strange(a_⌊n/6+1⌋, ..., a_⌊2n/6⌋, a_⌊4n/6+1⌋, ..., a_⌊5n/6⌋)
4.  for i = n down to 1 do
5.      print a_i
6.  strange(a_⌊2n/6+1⌋, ..., a_⌊3n/6⌋, a_⌊5n/6+1⌋, ..., a_n)
7.  strange(a_1, ..., a_⌊n/6⌋, a_⌊5n/6+1⌋, ..., a_n)
8.  strange(a_⌊2n/6+1⌋, ..., a_⌊4n/6⌋)
9.  for i = 1 to n do
10.     for j = 1 to i − 1 do
11.         if a_j > a_i then swap a_i and a_j
```

(a) [*5 marks*] Analyze the running time by giving a recurrence for this pseudocode. Show your work. (Remember to give tight Θ bounds. You may ignore floors and ceilings.)

Lines 4-5: $\Theta(n)$

Lines 9-11: $\Theta\left(\sum_{i=1}^{n}(i-1)\right) = \Theta(n^2)$

Lines 2,3,6,7,8: $5T(\frac{n}{3})$

$$\Rightarrow \quad T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 2012 \\ 5T(\frac{n}{3}) + \Theta(n^2) & \text{else} \end{cases}$$

(b) [*3 marks*] Solve your recurrence using the Master method.

$a=5$, $b=3$, $d = \log_3 5 \ll 2$, $f(n)=n^2$, $\varepsilon = 0.01$

Case 3, $f(n)/n^{d+\varepsilon}$ is increasing

$\Rightarrow T(n) = \Theta(n^2)$

2. [*30 marks*] *Short questions.*

(a) [*6 marks*] Arrange the following six functions in increasing order of growth rate. (Justifications are not required.) All logarithms are in base 2.

$$2^n, \ n^3, \ \frac{n^3(\log\log n)^3}{(\log n)^2}, \ n^{2.81}, \ n^{\log n}, \ \log(2012^n).$$

$\log(2012^n)$, $n^{2.81}$, $\frac{n^3(\log\log n)^3}{(\log n)^2}$, n^3, $n^{\log n}$, 2^n.

(b) [*3 marks*] State the recurrence for the running time of Karatsuba and Ofman's divide-and-conquer algorithm for multiplying two n-bit numbers.

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 3T(\frac{n}{2}) + \Theta(n) & \text{else} \end{cases}$$

(c) [*3 marks*] Describe a correct greedy strategy to solve the following *disjoint intervals* problem: given a set of n intervals $[a_1, b_1], \ldots, [a_n, b_n]$, choose a largest subset of intervals so that no two chosen intervals overlap. (No need to prove correctness.)

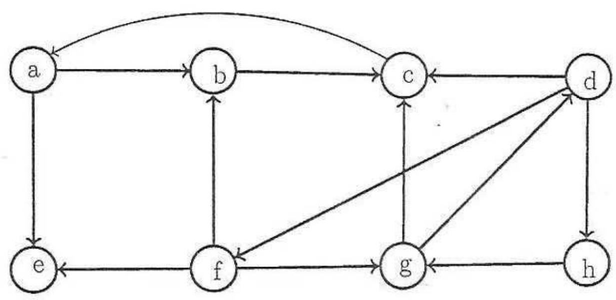
repeat {
pick the interval $[a_i, b_i]$ with smallest b_i
remove $[a_i, b_i]$ and all intervals intersecting it
}

(d) [*4 marks*] Consider the *coin changing* problem: given a set of n coin values and a target number W (all positive integers), find coins that sum to exactly W (assuming an unlimited supply of coins of each value). What is the running time of the correct (non-greedy) algorithm given in class for this problem, and is the running time considered to be truly polynomial in the input size? Explain.

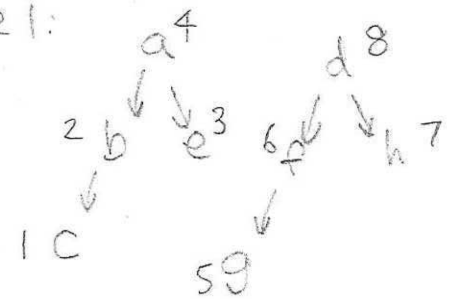
Runtime $O(nW)$

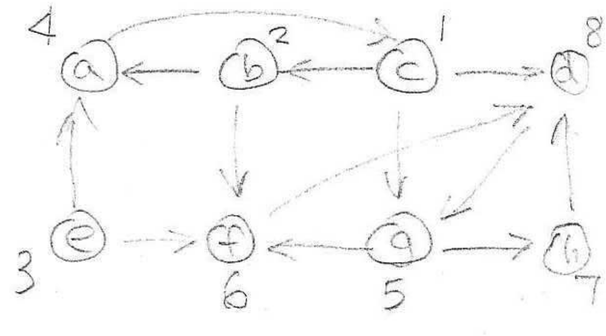
Not polynomial, since input size is $\Theta(n \log W)$ in # of bits.

(e) [*6 marks*] Run the algorithm for the *strongly connected components* problem from class (due to Sharir/Kosaraju). In the first pass, use alphabetical order; draw the DFS trees and number the vertices in order of finish. In the second pass, draw the transposed graph and the new DFS trees.

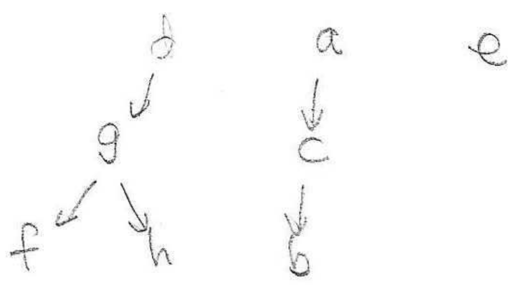


Phase 1:





Phase 2:



(f) [*4 marks*] Describe and compare the data structures required for an $O(m \log n)$-time implementation of Kruskal's and Prim's algorithms for *minimum spanning trees*.

Kruskal: need union/find data structure

Prim: need priority queue
(standard heap suffices to give $O(m \log n)$, or one could use Fibonacci heaps)

(g) [*4 marks*] For the *all-pairs shortest paths* problem for a weighted graph with n vertices and m edges, recall that repeated application of Dijkstra's algorithm can achieve $O(n^2 \log n + mn)$ running time while the Floyd–Warshall algorithm has $O(n^3)$ running time. Despite the apparently worse asymptotic runtime, describe two advantages of the Floyd–Warshall algorithm over Dijkstra's.

1. Smaller constant factors
(for dense graphs, F-W may be faster in practice)

2. works even when there are negative weights
(Dijkstra may be incorrect in this case)

3. [*16 marks*] *Dynamic programming.* Let $d(p, q)$ denote ~~the Euclidean distance of two points p and q~~ a cost function.

We are given a sequence of n points $p_1, \ldots, p_n$ in 2D, where each point is colored red or blue. We are also given a number k. We want to find a polygonal chain $\langle p_{i_0}, p_{i_1}, \ldots, p_{i_\ell} \rangle$ with $1 = i_0 < i_1 < i_2 < \cdots < i_{\ell-1} < i_\ell = n$, to minimize the total ~~length~~ cost $d(p_{i_0}, p_{i_1}) + d(p_{i_1}, p_{i_2}) + \cdots + d(p_{i_{\ell-1}}, p_{i_\ell})$, subject to the constraint that the number of *red* vertices on the chain is *at most* k (there is no constraint on the number of blue vertices used or the total number ℓ of vertices used).

Present an ~~$\Theta(nk)$~~ $O(n^2)$-time dynamic programming algorithm to solve the problem.

Definition of subproblems: $(i = 1, \ldots, n,\ j = 0, \ldots, k)$

$C[i, j]$ = min total cost over all chains from p_1 to p_i that have at most j red vertices

Base case(s):

$$C[1, j] = \begin{cases} \infty & \text{if } p_1 \text{ is red and } j = 0 \\ 0 & \text{else} \end{cases}$$

$$C[i, -1] = \infty$$

Recursive formula, with justifications:

$$C[i, j] = \begin{cases} \min_{\ell = 1, \ldots, i-1} \left(C[\ell, j-1] + d(p_\ell, p_i) \right) & \text{if } p_i \text{ is red} \\ \min_{\ell = 1, \ldots, i-1} \left(C[\ell, j] + d(p_\ell, p_i) \right) & \text{else} \end{cases}$$

(We take choices over the next-to-last vertex p_ℓ in the chain)

Initials: SOL'NS

Pseudocode for computing the optimal ~~length~~ cost:

```
1.   for j = 0 to k do
2.       if p1 is red and j = 0 then C[1,j] = ∞ else C[1,j] = 0
3    for i = 1 to n do C[i,-1] = ∞
4    for i = 1 to n do
5.       for j = 0 to k do {
6.           C[i,j] = ∞
7.           for ℓ = 1 to i-1 do
8.               if pi is red and C[ℓ,j-1] + d(pℓ,pi) < C[i,j]
9.                   C[i,j] = C[ℓ,j-1] + d(pℓ,pi), π[i,j] = ℓ
10.              if pi is blue and C[ℓ,j] + d(pℓ,pi) < C[i,j]
11                   C[i,j] = C[ℓ,j] + d(pℓ,pi), π[i,j] = ℓ
         }
12   return C[n,k]
```

Pseudocode for retrieving an optimal path:

```
     Retrieve(i,j):   // initially call Retrieve(n,k)
13.      if i=1 then print p1 and return
14       if pi is red then Retrieve(π[i,j], j-1)
15                    else Retrieve(π[i,j], j)
16       Print pi
```

Analysis:

Lines 1-3 $O(n)$ time

Lines 4-11 $O(n^2 k) \leq O(n^3)$

Lines 13-16 $O(k)$

Total time $O(n^3)$.

4. [*11 marks*] *Graph algorithms.*

(a) [*5 marks*] Let $G = (V, E)$ be a directed graph with n vertices and m edges. Consider the *Hamiltonian path* problem: decide whether there exists a path that visits every vertex exactly once. In the special case when G is a directed acyclic graph (DAG), describe how this problem can be solved in $O(m + n)$ time. Explain why your method is correct. (Hint: use a known algorithm for a specific problem we have studied from class for DAGs...)

1. Compute a topological sort $v_1, \ldots, v_n$ of G in $O(m+n)$ time by the alg'm from class (based on DFS)

2. return yes iff $v_1 v_2 \cdots v_n$ forms a path in G (which we can check easily in linear time)

Total runtime is $O(m+n)$.

Correctness:

if there is a Hamiltonian path $u_1, u_2, \cdots u_n$,
then by def'n of topological sort,
u_i must appear before u_{i+1} in the sort
i.e the sequence $u_1 u_2 \cdots u_n$ must be identical to $v_1 v_2 \cdots v_n$.

Initials: SOLNS

(b) [*6 marks*] Let $G = (V, E)$ be a weighted undirected graph with n vertices and m edges. Consider the following problem: given $s, t \in V$, find a path from s to t that minimizes the largest edge weight along the path. (This is the same problem considered in one of the questions from Assignment 4.) In the special case where all the edges in G have weights from $\{1, 2, \ldots, 2012\}$, describe how this problem can be solved in $O(m+n)$ time. (Hint: do not use minimum spanning tree algorithms, but instead use DFS or BFS. First solve the decision problem: given a value w, decide if the largest edge weight along the optimal path is at most w...)

To solve the decision problem, given value w:

consider the subgraph consisting of all edges with weight $\leq w$.

return "yes" iff s and t are connected in this subgraph

This takes $O(m+n)$ time by BFS or DFS.

To solve the original problem:

use linear search

for $w = 1$ to 2012

if decision algm says "yes" return w

Total time is $O(2012(m+n)) = O(m+n)$.

[Alternatively, we could use binary search.]

5. [*13 marks*] *More short questions on* P *and* NP.

(a) [*3 marks*] True or False: We know definitively that there is no polynomial-time algorithm for the CLIQUE problem. Briefly justify your answer.

False, since $P \neq NP$ has not been proved yet

[if student explicitly states $P \neq NP$ as an assumption, "True" is OK]

(b) [*3 marks*] True or False: There is a polynomial-time reduction from the SUBSET-SUM problem to the HAMILTONIAN-CYCLE problem. Briefly justify your answer.

True. SUBSET-SUM is in NP

HAMILTONIAN-CYCLE is NP-complete

$\Rightarrow$ by def'n of NP-completeness,

SUBSET-SUM $\leq_p$ HAMILTONIAN-CYCLE.

(c) [*3 marks*] Give a problem that, as we have shown in class, cannot be solved by an algorithm with running time less than $2^{2^{2^n}}$.

The halting problem. It can't be solved by <u>any</u> alg'm!

(d) [*4 marks*] Convert the following optimization problem into a decision problem in NP (you do not need to prove that the decision problem is in NP).

Input: a set P of m points, a set S of n rectangles in 2D, and an integer k.

Output: find a subset $T \subseteq S$ of k rectangles, maximizing the number of points in P that are covered by at least one rectangle in T.

Input: set P of m points, set S of n rectangles, integer k, integer ℓ

Output: yes iff $\exists$ subset $T \subseteq S$ of k rectangles, s.t. # of points in P covered by at least one rectangle in T is at least ℓ.

6. [*22 marks*] *Proving NP-completeness.* For two strings a and b, define the distance $d(a,b)$ to be the number of positions j such that the j-th symbol of a does not match the j-th symbol of b. For example, $d(00110, 11000) = 4$.

Consider the following problem DISTANT-STRING:

> *Input*: a finite set Σ, a collection of m strings $a_1, \ldots, a_m$ where each a_i is a string of n symbols from Σ, and an integer K.
> *Output*: "yes" iff there exists a string b of n symbols from ~~Σ~~ $\{0,1\}$, such that $d(a_i, b) \geq K$ for all $i = 1, \ldots, m$.

For example, for input $\Sigma = \{0,1\}$, $a_1 = 00110$, $a_2 = 00111$, $a_3 = 10111$, and $K = 4$, the output is yes, by choosing $b = 11000$.

(a) [*3 marks*] For the input $\Sigma = \{0,1,2\}$, $a_1 = 220002$, $a_2 = 221020$, $a_3 = 222100$, $a_4 = 221211$, and $K = 4$, is the answer "yes"? Justify your answer.

Yes. e.g. pick b = 001110.

[all possible choices for b:

XX0010 XX0101 XX1001 XX1101
XX0011 XX0110 XX1110
XX0111]

(b) [*5 marks*] Prove that DISTANT-STRING is in NP.

Certificate: string b, which has polysize (n bits)
Condition to verify: $\forall i = 1, \ldots, m$, $d(a_i, b) \geq K$
which can be done in polytime
since computing $d(a_i, b)$ takes linear time for each i

(c) [*14 marks*] Prove that DISTANT-STRING is NP-complete, using the known fact that 3SAT is NP-complete.

- Give a polynomial-time reduction from 3SAT to DISTANT-STRING.

Given:

3CNF formula F, say with n vars $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$

To construct:

set Σ, m strings $a_1, \ldots, a_m$, and integer K

The construction: (Remember to check that it runs in polytime. Hint: create strings similar to the example from part (a)...)

$\Sigma = \{0, 1, 2\}$

for $i = 1, \ldots, m$, for $j = 1, \ldots, n$,

$$j^{th} \text{ bit of } a_i = \begin{cases} 0 & \text{if } x_j \text{ appears in } C_i \\ 1 & \text{if } \overline{x}_j \text{ appears in } C_i \\ 2 & \text{else} \end{cases}$$

$K = n - 2$.

This construction clearly takes polytime.

SOL'NS

- Prove the correctness of your reduction.

To show: F is satisfiable $\iff$ $\exists$ string $b \in \{0,1\}^n$ s.t. $\forall i$ $d(a_i, b) \geq K$.

Proof (two directions):

($\Rightarrow$) Suppose F has a satisfying assignment α.

Let jth bit of $b = \begin{cases} 1 & \text{if } x_j \text{ is true in } \alpha \\ 0 & \text{else.} \end{cases}$

Then $\forall i$,

a_i and b match in position j

iff x_j is in C_i and x_j is false, or

$\bar{x}_j$ is in C_i and x_j is true.

this cannot happen 3 times since α satisfies C_i

$\Rightarrow$ a_i and b match in ≤ 2 positions

$\Rightarrow$ $d(a_i, b) \geq n-2 = K$.

($\Leftarrow$) Suppose there is such a string b.

Define assignment α: if jth bit of b is 1, set x_j true else set x_j false.

Then $\forall i$,

$d(a_i, b) \geq n-2 = K$

$\Rightarrow$ a_i and b match in ≤ 2 positions

if x_j is in C_i and x_j is false, or

$\bar{x}_j$ is in C_i and x_j is true,

then a_i and b match in position j

this cannot happen 3 times

$\Rightarrow$ α satisfies C_i