

SI433, Practice Problems

Far and away the most important thing for you to study is the first two exams. Second would be the lecture notes, homeworks, and project from after the 12-week exam.

1. Simplify the following expressions:

(a) $O(3n^3 + n^2 \lg n)$

(b) $O(n + n \lg n) + \Theta(n^2)$

(c) $\Theta(n + n \lg n) + O(n^2)$

2. (? Points) Suppose Algorithm A and Algorithm B are two algorithms for solving the shortest path problem, A runs in $\Theta(n^2)$ time (worst case), and B runs in $\Theta(n \lg n)$ time (worst case). Is it true that for any graph and any two vertices A will take more time than B?

3. (? Points) MIDN Jones analyzes an algorithm and determines that its best case complexity is $O(n^2)$. MIDN Smith analyzes the same algorithm and determines that its best case complexity is $\Theta(n \lg n)$. Could they both be right, or must at least one of them be wrong? Justify your answer!

4. (? Points) Give the worst case time complexity of the following algorithm:

```
void alg(A,n)
{
    for(int i = 0; i < n; i++)
    {
        for(int j = 1; j < n - i; j++)
        {
            if (A[j-1] > A[j])
            {
                t = A[j];
                A[j] = A[j-1];
                A[j-1] = t;
            }
        }
    }
}
```

5. (? Points) Suppose A is an array of n ints, all in the range $1, 2, \dots, k$, and suppose that `mystery(M)`, where M is an `int`, is a function with best case time complexity $\Theta(M \lg M)$ and worst case time complexity $\Theta(M^2)$. What can you tell me about the time required by the following algorithm in terms of n and k ?

```
int sum = 0;
for(int i = 0; i < n; i++)
{
    int x = mystery(A[i]);
    sum += x;
}
cout << sum << endl;
```

I'll start you off: The time is $O(2^{(n+k)})$ and $\Omega(1)$. Improve on this by giving me a more precise answer! Justify your answers!

6. (? Points) Consider the following algorithm, which determines whether $1 + 2 + \dots + n$ is a perfect square:

```
bool test(int n)
{
    for(int i = 1; i <= n; i++)
    {
        // Compute sum = 1 + 2 + ... + n
        int sum = 0;
        for(int k = 1; k <= n; k++)
            sum = sum + k;

        // Compare sum and i^2
        if (sum == i*i)
            return true;
    }
    return false;
}
```

- (a) (10 Points) In terms of n , what is the worst case time complexity of this algorithm? (Justify your answer!)
- (b) (5 Points) Modify the algorithm to improve its complexity (it still has to compute the same thing!) and analyze the complexity of your improved algorithm.

7. (? Points) Consider the following algorithm, which returns true if a zero appears in array A in the range i to j:

```
bool zeros(int *A, int i, int j)
{
    // BASE CASE
    if (i == j)
        return A[i] == 0;

    // DIVIDE
    int k = (i + j)/2;

    // CONQUER AND COMBINE
    if(zeros(A,i,k))
        return true;
    if(zeros(A,k+1,j))
        return true;
    return false;
}
```

- (a) (10 Points) Give a recurrence relation for the worst case time complexity for this algorithm. Justify your answer!
- (b) (5 Points) Give a recurrence relation for the best case time complexity for this algorithm. Justify your answer!
8. (? Points) What is the growth rate of the function $T(n)$ defined by $T(n) = 3T(n/2) + \Theta(n)$?
9. (? Points) Which sorting algorithm is best and which is worst? Defend your answers!
10. (? Points) Draw the graph defined by the following adjacency matrix:

	v_0	v_1	v_2	v_3	v_4
v_0	0	0	1	0	1
v_1	0	0	1	0	0
v_2	1	0	0	0	0
v_3	1	1	0	0	0
v_4	1	0	1	0	0

11. (? Points) In terms of n , the number of vertices in graph G , and e , the number of edges, what is the complexity of the following algorithm if G is represented using an *adjacency matrix*?

```
BFS(G,i): G is the graph, i is the source vertex
{ initialize color to white for all vertices
  color(i) = gray
  let Q be a queue of vertices (initially empty!)
  Q.enqueue(i)
  while(!Q.empty())
  { v = Q.dequeue()
    for each neighbor w of v do
      if color(w) = white
        color(w) = gray
        Q.enqueue(w)
    color(v) = black
  }
}
```

12. (? Points) The change making problem is as follows: Given coin denominations D_1, D_2, \dots, D_k (and an infinite supply of each type of coin) give a customer change for X cents using as few coins as possible.

I've developed a greedy algorithm for solving this problem, which is:

- (a) if $X = 0$ stop
- (b) choose D_i the largest denomination not exceeding X
- (c) hand the customer a coin of denomination D_i , and set $X = X - D_i$
- (d) goto Step (a)

Prove that this greedy algorithm is not optimum, meaning it doesn't always make change with the fewest coins. (Hint: It actually does work for denominations 25,10,5,1, which is pretty handy, since those are the denominations we use!)

13. (? Points) What do finite automata and expression trees have to do with Huffman encoding?

14. (? Points) The gcd of 252 and 147 is 21. Use the extended gcd algorithm to compute two numbers s and t such that $s \cdot 252 + t \cdot 147 = 21$. Show your work!

```
(g, s, t) ← egcd(u, v)
{
  if v = 0 return (u, 1, 0)
  q = quotient of u divided by v
  r = remainder of u divided by v
  (g', s', t') = egcd(v, r)
  return (g', t', s' - t'q)
}
```

15. (? Points) What would you have to do in order to forge a message with out Project 3 electronic signature format?
16. (? Points) Why is `power(m,e) % n` a lousey way to compute $m^e \bmod n$, even when `power` is a very efficient integer exponentiation algorithm?
17. (? Points) What does *NP* stand for?
18. (? Points) Prove that the Hamiltonian Cycle problem is in NP.
19. (? Points) Why would solving an NP-Complete problem prove that $P = NP$?
20. (? Points) Give a polynomial time reduction of the set-partition problem to the 0/1 Knapsack problem. Note that this is the reverse of what we did in class!
21. (? Points) Why didn't I insist on you guys getting the longest path possible for Project 2?
22. (? Points) What's the difference between saying "operation X runs in $O(\lg n)$ time" and "operation X runs in $O(\lg n)$ amortized time"?