Matthew Stroble

Missouri University of Science & Technology — Department of Computer Science

Fall 2018 — **CS 2500: Algorithms**

Homework 6: Final Exam Handout

Instructor: *Sid Nadendla* — **Due:** *December 11, 2018*

This homework set has problems that span across the topics covered in this entire course, and should be used as one of the handouts for the final exam. In addition to the questions in this homework assignments, students should also study their midterm handouts. The final exam will be 1.5hr long, with 2 large problems (from midterm handouts) and 2-3 small problems (from this problem set).

Problem 1: *20 points*

Map the definitions of the following asymptotic notations:

a. $f(n) = O(g(n))$ — i. $c_1 g(n) \leq f(n) \leq c_2 g(n),\ \forall\, n \geq n_0$

b. $f(n) = \Omega(g(n))$ — ii. $0 \leq f(n) \leq cg(n),\ \forall\, n \geq n_0$

c. $f(n) = \Theta(g(n))$ — iii. $0 \leq cg(n) \leq f(n),\ \forall\, n \geq n_0$

d. $f(n) = o(g(n))$ — iv. $0 \leq cg(n) < f(n),\ \forall\, n \geq n_0$

e. $f(n) = \omega(g(n))$ — v. $0 \leq f(n) < cg(n),\ \forall\, n \geq n_0$

Problem 2: *10 points*

Consider the following program which returns the maximum entry in a given array A:

```
RMAX(A)
n = A.length                      1
est_max = Rmax(A[2 : n])          T(n-1)
if A[1] ≥ est_max                 1
    return A[1]                   1
    return est_max                1
```

If n is the size of the input array, the recursion in the run-time of this algorithm is given by

1. $T(n) = T(n-1) + O(n)$
2. $T(n) = n \cdot T(n-1) + O(1)$

k

3. $T(n) = T(n-1) + O(1)$
4. $T(n) = T(n/2) + O(1)$

Problem 3: *20 points*

Map the worst-case run-times to the algorithms presented in the following table.

a. Heap Sort	i. $O(n^2)$
b. Insertion Sort	ii. $O(n \log n)$
c. Quick Sort	iii. $O(n^2)$
d. Radix Sort	iv. $O(n^2)$
e. Merge Sort	v. $O(n)$

$O(nk)$

Problem 4: *5 points*

An optimization problem can be solved using dynamic programming if _____ can be derived from the multi-stage representation of its value.

1. Bellman equation
2. a greedy algorithm
3. asymptotic upper bound
4. all of the above

Problem 5: *5 points*

A greedy algorithm returns an optimal solution to an optimization problem if there exists

1. an optimal substructure within the optimization problem
2. greedy-choice where locally optimal solutions accumulate to a globally optimal solution
3. all of the above
4. none of the above

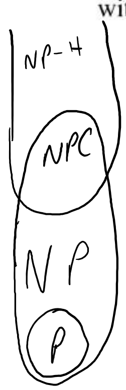
Problem 6: ***20 points***

Map the algorithms listed in the left column, to the applications listed in the right column within the following table.

a. Breadth-First Search	i. Detect negative-weight cycles
b. Depth-First Search	ii. Tracing/Removing Garbage (Memory)
c. Prim's Algorithm	iii. GPS Navigation
e. Bellman-Ford's Algorithm	iv. Traffic-Flow Control
e. Dijkstra's Algorithm	v. Circuit Design
f. Edmond-Karp's Algorithm	vi. Loop Detection

Problem 7: ***20 points***

Map the problems listed in the left column, to the complexity classes listed in the right column within the following table. Show all the appropriate mappings, if there are more than one.



a. 0-1 Knapsack	i. P
b. Halting Problem	ii. NP
c. Generalized Chess	iii. NP-Hard
e. Traveling Salesman	iv. NP-Complete
e. Satisfiability	v. EXP
f. Shortest Path	vi. Unsolvable