Practice for Test 2

Advanced Algorithms I (7081) Fall 2015

*The questions on Test 2 on Wednesday, Nov 25 will be similar to a **subset** of questions from this Practice. It is based on material we covered in class from Chapters 5, 6, 7, 8 of the text.*

PART I: Multiple Choice: For each of the following questions **circle** the **best** answer

1. Best algorithm for computing shortest (minimum hop-length) paths in (unweighted) digraphs:
    a) DFS
    b) BFS
    c) Topological sort
    d) Prim's algorithm
    e) Dijkstra's algorithm
2. Best algorithm for computing shortest (minimum-weight) paths in weighted digraphs:
    a) DFS
    b) BFS
    c) Topological sort
    d) Prim's algorithm
    e) Dijkstra's algorithm
3. An algorithm that can be used to test whether a graph is connected:
    a) BFS but not DFS
    b) DFS but not BFS
    c) Either BFS or DFS can be used
    d) Topological sort
    e) None of the above
4. An algorithm that can be used to compute the diameter of a graph:
    a) BFS
    b) DFS
    c) Either BFS or DFS can be used
    d) Topological sort
    e) None of the above
5. Strassen's algorithm is based on the following key result and design strategy:
    a) The product of two 2x2 matrices can be computed using 8 multiplications; Backtracking
    b) The product of two 2x2 matrices can be computed using 8 multiplications; Divide-and-conquer
    c) The product of two 2x2 matrices can be computed using 7 multiplications; Greedy Method
    d) The product of two 2x2 matrices can be computed using 7 multiplications; Divide-and-conquer
    e) The product of two nxn matrices can be computed using n^3 multiplications; Divide-and-conquer
6. The algorithm for computing Huffman trees utilizes the design strategy:
    a) Backtracking
    b) Divide-and-conquer
    c) Greedy Method
    d) Dynamic Programming
    e) Adhoc
7. An efficient way to test for cycles when implementing Kruskal's algorithm is:
    a) BFS
    b) Disjoint sets ADT with union and find operations
    c) Dynamic programming
    d) Backtracking
    e) Min-heap

8. Dijkstra's algorithm for computing shortest paths in digraphs utilizes the design strategy:
    a) Backtracking
    b) Divide-and-conquer
    c) Greedy Method
    d) Dynamic Programming
    e) Adhoc
9. Strassen's algorithm for matrix multiplicationis based on the following key result and design strategy:
    a) The product of two 2x2 matrices can be computed using 8 multiplications; Backtracking
    b) The product of two 2x2 matrices can be computed using 8 multiplications; Divide-and-conquer
    c) The product of two 2x2 matrices can be computed using 7 multiplications; Greedy Method
    d) The product of two 2x2 matrices can be computed using 7 multiplications; Divide-and-conquer
    e) The product of two nxn matrices can be computed using n^3 multiplications; Divide-and-conquer
10. Floyd's algorithm for computing all-pairs shortest paths utilizes the design strategy:
    a) Backtracking
    b) Divide-and-conquer
    c) Greedy Method
    d) Dynamic Programming
    e) Adhoc
11. Assume a weighted graph G on n vertices is implemented using its adjacency matrix. The worst-case complexity of Kruskal's algorithm is:
    a) $O(\log n)$
    b) $O(n)$
    c) $O(n \log n)$
    d) $O(n^2)$
    e) $O(n^2 \log n)$
12. Assuming the graph G on n vertices and m edges is implemented using its **adjacency matrix** and an **array** is used to implement the priority queue used to choose next node to be added to the tree then Prim's minimum spanning tree algorithm has worst-case complexity:
    a) $\Theta(n^2)$
    b) $\Theta(n^3)$
    c) $\Theta(n \log n)$
    d) $\Theta(m \log n)$
    e) $\Theta(mn)$
13. Assuming a graph G on n vertices and m edges is implemented using **adjacency lists** and a **min-heap** is used to implement the priority queue used to choose the next node to be added to the tree. Prim's minimum spanning tree algorithm has worst-case complexity:
    a) $\Theta(n^2)$
    b) $\Theta(n^3)$
    c) $\Theta(n \log n)$
    d) $\Theta(m \log n)$
    e) $\Theta(mn)$
14. Assuming a digraph D on n vertices and m edges is implemented using its **adjacency matrix** and an **array** is used to implement priority queue used to choose the next node to be added to the tree, Dijkstra's shortest path algorithm has worst-case complexity:
    a) $\Theta(n^2)$
    b) $\Theta(n^3)$
    c) $\Theta(n \log n)$
    d) $\Theta(m \log n)$
    e) $\Theta(mn)$

15. Assuming a digraph D on n vertices and m edges is implemented using **adjacency lists** and a **min-heap** is used to implement the priority queue used to choose next the node to be added to the tree, Dijkstra's shortest path algorithm has worst-case complexity:
    a) $\Theta(n^2)$
    b) $\Theta(n^3)$
    c) $\Theta(n \log n)$
    d) $\Theta(m \log n)$
    e) $\Theta(mn)$
16. The number of multiplications performed by Strassen's algorithm to compute the product of two 4×4 matrices :
    a) 10
    b) 16
    c) 36
    d) 49
    e) 64
17. The greedy solution to the knapsack problem is based on sorting the objects in:
    a) increasing order of their values
    b) decreasing order of their values
    c) increasing order of their densities (value/weight)
    d) decreasing order of their densities (value/weight)
    e) decreasing order of their weights
18. Two primitive 4^{th} roots of unity are:
    a) 1 and -1
    b) -1 and `i`
    c) -1 and `-i`
    d) `i` and `-i`
    e) none of the above
19. *FFT* has worst-case complexity $W(n)$ given by
    a) $\Theta(\log n)$
    b) $\Theta((\log n)^2)$
    c) $\Theta(n)$
    d) $\Theta(n \log n)$
    e) $\Theta(n^2)$
20. The worst-case complexity of Floyd's All-Pairs Shortest Path algorithm is:
    a) $\Theta(\log n)$
    b) $\Theta(n)$
    c) $\Theta(n \log n)$
    d) $\Theta(n^2)$
    e) $\Theta(n^3)$

PART II: Explain your answers in detail.

1. a) Name four major algorithm design strategies and briefly describe them.
   b) Divide-and-conquer is a _____ recursive strategy, whereas dynamic programming is a _____ recursive strategy

2. Does the Greedy method, i.e., choosing the most of the largest denomination coin at each stage, hold for American denomination coins? Does it hold for American denomination when there are no nickels?

3. Solve the following instance of the Knapsack problem, i.e., give fraction of each object chosen and value of optimal Knapsack. Show steps:

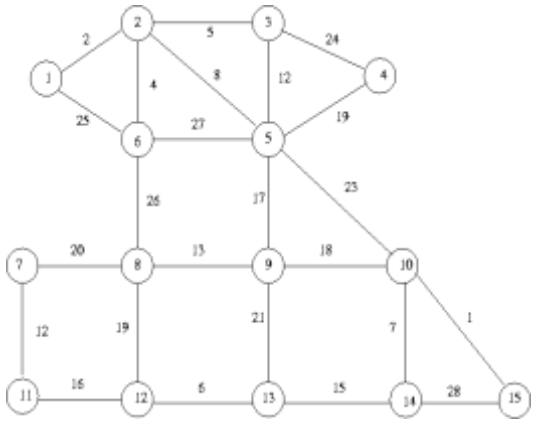
Capacity of Knapsack is $C = 100$

Object	B0	B1	B2	B3	B4	B5
Value	30	200	100	50	10	1000
Weight	10	100	100	10	1	2000

4. Obtain the Huffman tree and associated Huffman code for the following symbols and frequencies:

Symbol	a	b	c	d	e	f	g
Frequency	10	3	1	5	5	2	4

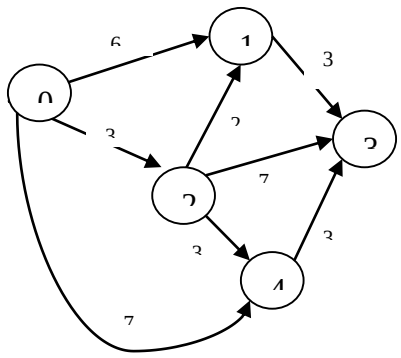
5. a) Show the action of Kruskal's algorithm for the graph below. Mark the edges of the minimum spanning tree and label them in the order they are chosen.



b) Repeat part a) for Prim's algorithm starting at vertex 1.

6. Explain how to detect cycles efficiently in Kruskal's algorithm. Describe the data structure for this and how it is applied.

7. Show the action of Dijkstra's algorithm for the digraph below with root vertex 0.



8. a) Suppose $P(x)$ and $Q(x)$ are two polynomials of (even) size n. Let $P_1(x)$ and $P_2(x)$ denote the polynomials of size n/2 determined by the first $n/2$ and last n/2 coefficients of $P(x)$. Similarly define $Q_1(x)$ and $Q_2(x)$, i.e., $P = P_1 + x^{n/2} P_2$. and $Q = Q_1 + x^{n/2} Q_2$. Show how the product PQ can be computed using only 3 ***distinct*** multiplications of polynomials of size $n/2$.

b) Briefly explain how the result in a) can be used to design a divide-and-conquer algorithm for multiplying two polynomials of size n (explain what the recursive calls are and what the bootstrap condition is).

c) Analyze the worst-case complexity of algorithm you have given in part b). In particular **derive** a recurrence formula for $W(n)$ and **solve**. As usual, to simplify the math, you may assume that n is a power of 2.

9. Describe the design of a divide-and-conquer algorithm for multiplying two $n \times n$ matrices (Strassen's Algorithm) based on the fact that the product of two 2×2 matrices can be computed using only 7 multiplications.

10. a) Derive the recurrence relation for the worst-case complexity $W(n)$ of Strassen's algorithm for computing the product of two matrices A and B of size n and solve to obtain a formula for $W(n)$.

11. Explain how the n^{th} power of an nxn matrix can be computed in time $O(n^{\log_2 7} \log n)$. Name algorithms and how they are used.

12. **Derive** the recurrence relation for the worst-case complexity $W(n)$ of *FFT* and **solve** to obtain a formula for $W(n)$.

13. For $n = 2^k$, let ω be a primitive n^{th} root of unity, and let $\alpha = \omega^2$ be a primitive $(n/2)^{\text{th}}$ root of unity. Let $P(x) = a_0 + a_1 x + \ldots a_{n-1} x^{n-1}$, $Q(x) = DFT_\omega(P(x)) = b_0 + b_1 x + \ldots + b_{n-1} x^{n-1}$,

$E(x) = a_0 + a_2 x + \ldots + a_{n-2} x^{n/2-1}$, $D(x) = a_1 + a_3 x + \ldots a_{n-1} x^{n/2-1}$,

$DFT_\alpha(E(x)) = e_0 + e_1 x + \ldots + e_{n/2-1} x^{n/2-1}$, $DFT_\alpha(D(x)) = d_0 + d_1 x + \ldots + d_{n/2-1} x^{n/2-1}$.

a) Give a formula for ω using cos and sin:

b) Give a formula for b_j using $P(x)$ and ω:

c) Give a formula for b_j ($j = 0, \ldots, n-1$) in terms of e_j and d_j ($j = 0, \ldots, n/2 - 1$):

d) Let $c_0, c_1, \ldots, c_{63}$ be the coefficients in leaf nodes of the tree of recursive calls for *FFT* (read from left to right) with $n = 128$. Compute j for $a_j = c_{31}$. Explain.

14. Show the action of *FFT* for the polynomial $P(x) = 2x^3 + 3x - 2$

15. a) Using a commutative diagram show how *FFT* can be used to obtain an $O(n \log n)$ algorithm multiplying the two polynomials:

b) Demonstrate part a) for the two polynomials:

$$3x^2 + x - 2 \quad \text{and} \quad 6x + 1.$$

16. a) State the *Principle of Optimality* from Dynamic Programming.
b) Show the Principle of optimality holds for shortest paths

17. Consider Floyds's All-Pair Shortest Path algorithm. Derive the recurrence relation on which the design of the algorithm is based. Don't forget the initial condition. Define all notation that you use and EXPLAIN your derivation.

18. Draw and show the action of Floyd's algorithm for the weighted digraph below:

$V = \{0,1,2,3\}$, $E = \{(0,1),(0,3),(1,2),(2,1),(2,3),(3,0),(3,1)\}$ such that

w(0,1) = 1, w(0,3) = 4, w(1,2) = 1, w(2,1) = 6, w(2,3) = 1, w(3,0) = 2, w(3,1) = 4