Ph.D. Qualifying Examination

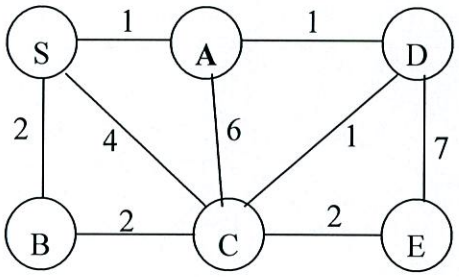
Fall 2011

Design and Analysis of Algorithms

Question II (100 points)

Part 1: (40pts)

Suppose Dijkstra's algorithm is run on the following undirected graph G, starting from the vertex *S*.

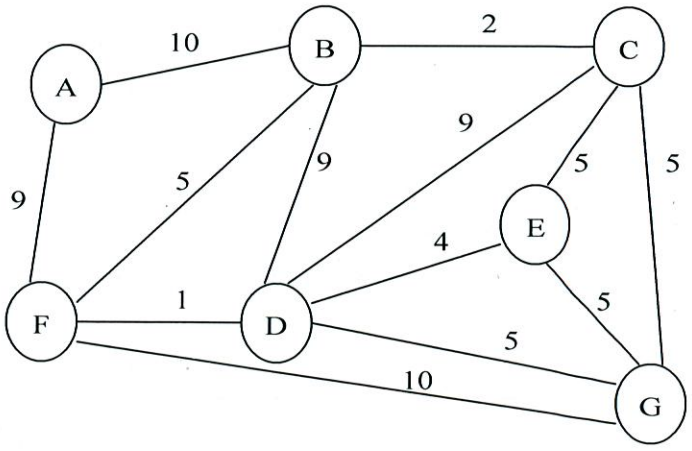


a. Draw a table showing the intermediate distance values of all the vertices after each iteration (20 pts)

b. Following question (a), please draw the final shortest-path tree (20 pts)

Part 2: (30pts)

Run Prim's algorithm on the following graph. Starting from vertex A, fill out the table showing the intermediate values of the cost and the previous node. (Whenever there is a choice of vertices, always use the alphabetical ordering).



(continued next page)

Set S	A Cost/prev	B Cost/prev	C Cost/prev	D Cost/prev	E Cost/prev	F Cost/prev	G Cost/prev

Part 3: (30 pts) Determine a Longest Common Subsequence (LCS) between X="10100101" and Y="110010101" (excluding double quotes). Please show your work.

Ph.D. Qualifying Examination

Fall 2011

Design and Analysis of Algorithms

Question I (100 points)

50 points

(1) Find an asymptotically tight bound for the recurrence:

$$T(n) = T(\sqrt[3]{n}) + (\log_2 n)^2$$

Show your work.

50 points

(2) Let A[1..n] be a sorted array, which means A[i] < A[j] for any i < j. Assume that all the numbers in the array are distinct. The problem asks to search for a specific value v in the array A. If such a value is found, then return the index i for which A[i] = v. Otherwise, return a message "not found". Write the pseudocode of an algorithm that performs the search operation in $O(\log_2 n)$ worst case running time. Argue why the worst case running time of your algorithm is $O(\log_2 n)$.

PhD Qualifying Examination

Spring 2010

Design and Analysis of Algorithms

Question I 100 points, to be graded as a whole (rather than having each part apportioned a specific number of points in advance).

(1) List the following functions from lowest order to highest order. If any two (or more) are of the same order, indicate which.

$5n + \lg n$ $(1+n)\lg n$ $3n^2 - 7n$ $\lg n$ 2^n $2^{10}n$ $4^{\lg n}$

(n+1)! $\sqrt{\lg n}$ $\lg \lg n$ n $n - n^3 + 7n^5$ $\sqrt{n}$ $n^2 + \lg n$

$\lg n^2$ n^3 e^n

(2) Given the functions in part (1):

Which belong to:

(A) $O(n^2)$

(B) $\Theta(n^2)$

(C) $\Omega(n^2 \lg n)$

(You do not have to show your work.)

(3) Reduce this recurrence relation to closed form, where $T(1) = 1$. Show all your work.

$T(n) = T(n/2) + \lg n$

Ph.D. Qualifying Examination

Spring 2010

Design and Analysis of Algorithms

Question II (100 points) The entire question will be graded as a whole.

1) Prove that for all even numbers x (where x is greater than 2) there exists an undirected connected graph where each node is connected to other nodes by exactly 3 edges.

For all of the following problems assume that a connected undirected graph G is implemented using an adjacency list.

2) Under what circumstances would it be preferable to implement the graph using an adjacency matrix?

3) Under what circumstances could the minimum spanning tree of G have fewer nodes than G?

4) Write pseudocode to efficiently find the minimum spanning tree of G. What is the name of the algorithm that you used?

Ph.D. Qualifying Examination
Spring 2009
Design and Analysis of Algorithms

Question I (100 points)

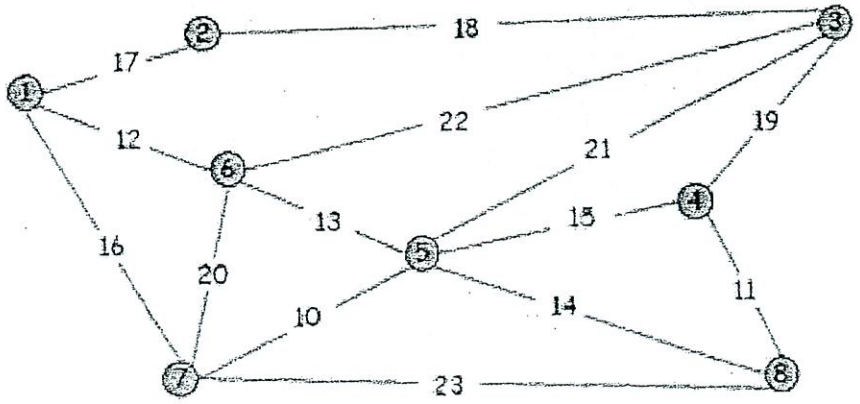
Part 1: (50 pts)

Suppose $A=\{a_1, a_2, a_3,\ldots,a_k\}$ is a set of distinct coin values (all the a_i are positive integers, and assume $a_1=1$) available in a particular country. The *coin changing problem* is defined as follows: Given an integer n, find the minimum number of coins from A that add up to n (assuming that all coins have value in A). Using dynamic programming to design a $O(nk)$ algorithm to solve the coin changing problem, that is, given inputs A and n, you algorithm outputs the minimum number of coins in A to add up to n (It is not necessary to say what the coins are. Prove your solution is correct).

For example, if, $A=\{1, 5, 8\}$ and $n=26$, the best way of making change uses 4 coins, *i.e.*, $\{5, 5, 8, 8\}$

Part 2: (25 pts)

In the following undirected graph, start from vertex 1, please list the edges that Prim's Minimum Spanning Tree algorithm adds to the solution (please list in the order in which they are added). [Write an edge as (s, y), for instance.] Show your work.



Part 3: (25 pts)

In the above undirected graph, please list the edges that Kruskal's Minimum Spanning Tree algorithm adds to the solution (please list in the order in which they are added). [Write an edge as (s, y), for instance.] Show your work.

Ph.D. Qualifying Examination

Spring 2009

Design and Analysis of Algorithms

Question II (100 points)

50 points

(1) Find an asymptotically tight bound for the recurrence:

$T(n) = 8T(\sqrt{n}) + (\log_2 n)^2$

Show all your work.

50 points

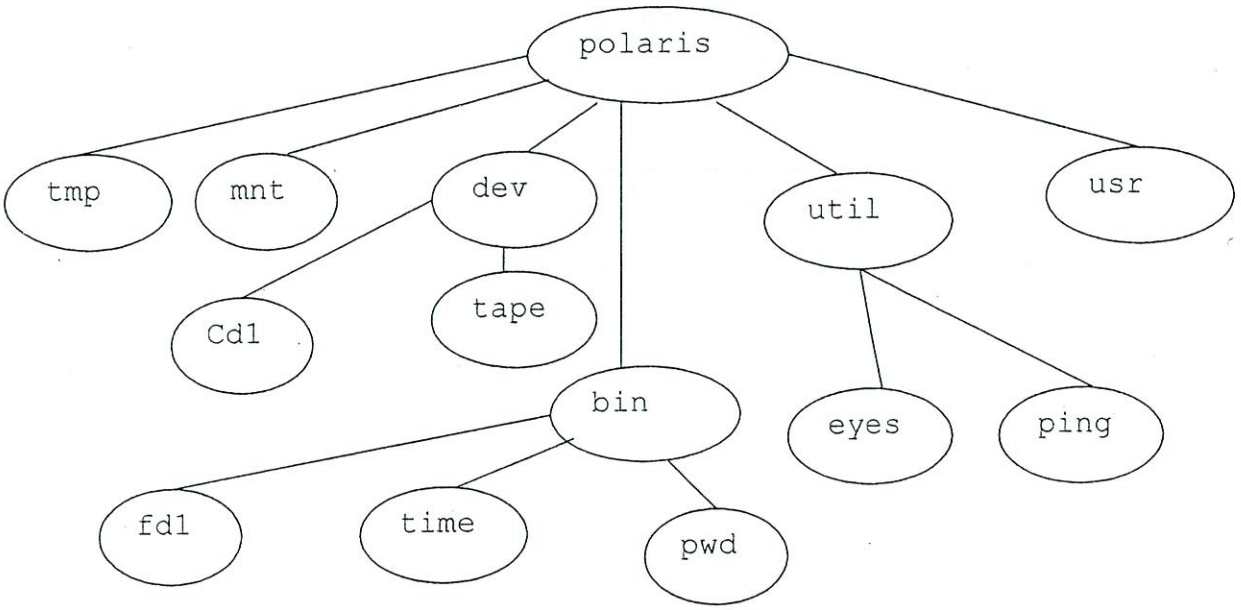
(2) Let A[1..n] and B[1..n] be two arrays of integers, such that each element of A or B is in the range *0* to m, for some integer $m = O(n)$. Design an $O(n)$ algorithm that finds two elements A[i] and B[j] such that A[i] + B[j] = k, where k is given. If no two such elements exist, then return an error message.

Ph.D. Qualifying Examination

Spring 2008

Data Structures and Algorithm Analysis

Question I (100 points)
Let T be the tree given below:



Assume the following functions belong to a complete working system.

Part A (50 points)
Using the tree T above, provide the output of the following function, firstPrint(T, T.root())

```
void firstPrint(const Tree& T, const Pos& v)
{
     cout << v.element();
     PosIterator children = T.children(v);
     while (children.hasNext()) {
          cout << " ";
          firstPrint(T, children.next());
     }
}
```

Part B (50 points)
Using the tree T above, provide the output of the following function, secondPrint(T, T.root())

```
void secondPrint(const Tree& T, const Pos& v)
{
     PosIterator children = T.children();
     while (children.hasNext()) {
          secondPrint(T, children.next());
          cout << " ";
     }

     cout << v.element();
}
```

Ph.D. Qualifying Examination

Spring 2008

Design and Analysis of Algorithms

Question II (100 points)

50 points

(1) Find an asymptotically tight bound for the recurrence:

$T(n) = 5T(n/2) + n^3 \log_2 n$

Show all your work.

50 points

(2) Use pseudocode to design a $\Theta(n)$-time nonrecursive procedure that reverses a singly linked list of n elements. The procedure should use no more than constant storage beyond that needed for the list itself.

PhD Qualifying Examination

Fall 2007

Design and Analysis of Algorithms

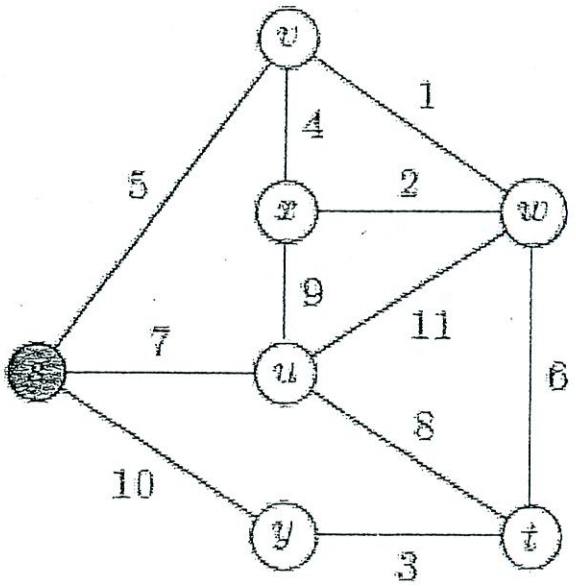
Question I (100 points)

Part 1: (60pts, 30pts/each)

In the following undirected graph G:

a. Give the first four edges that Prim's Minimum Spanning Tree algorithm (starting from s) adds, in the order in which they are added. [Write an edge as (s, y), for instance.] Show your work.

b. Give the first four edges that Dijkstra's Shortest Path algorithm (starting from s, going to t) adds, in the order in which they are added. Here, by "adding an edge" (x, y), we would mean that the algorithm sets the predecessor pointer of y to be x. Show your work.



Part 2: (50 pts) Consider a weighted directed acyclic graph G(V, E) and a source vertex *s*. Design an $\Theta(V+E)$ algorithm that computes the longest paths from *s* to all other vertices in G.

PhD Qualifying Examination

Spring 2007

Design and Analysis of Algorithms

Question I 100 points, to be graded as a whole (rather than having each part apportioned a specific number of points in advance).

(1) List the following functions from lowest order to highest order. If any two (or more) are of the same order, indicate which.

$5n + \lg n$ $\quad (1+n)\lg n$ $\quad 3n^2 - 7n$ $\quad \lg n$ $\quad 2^n$ $\quad 2^{10}n$ $\quad 4^{\lg n}$

(n+1)! $\quad \sqrt{\lg n}$ $\quad \lg\lg n$ $\quad n$ $\quad n - n^3 + 7n^5$ $\quad \sqrt{n}$ $\quad n^2 + \lg n$

$\lg n^2$ $\quad n^3$ $\quad e^n$

(2) Given the functions in part (1):

Which belong to:

(A) $O(n^2)$

(B) $\Theta(n^2)$

(C) $\Omega(n^2 \lg n)$

(You do not have to show your work.)

(3) Reduce this recurrence relation to closed form, where $T(1) = 1$. Show all your work.

$T(n) = T(n/2) + \lg n$

Ph.D. Qualifying Examination

Spring 2007

Design and Analysis of Algorithms

Question II (100 points)

50 points

(1) Given an undirected and connected graph in which all edges have the same weight, describe an algorithm to compute a minimum spanning tree in O(E) time.

50 points

(2) Consider a weighted directed acyclic graph G(V, E) and a source vertex *s*. Design an $\Theta(V+E)$ algorithm that computes the longest paths from *s* to all other vertices in G.

Ph.D. Qualifying Examination

Fall 2007

Design and Analysis of Algorithms

Question II (100 points)

50 points

(1) Find an asymptotically tight bound for the recurrence:

$T(n) = 3T(\sqrt{n}) + 100$

Show all your work.

50 points

(2) The operation HEAP-DELETE(A, i) deletes the item in node i from max-heap A. Give a pseudocode implementation of the HEAP-DELETE operation that runs in $O(\log_2 n)$ time for an n-element max-heap.

Ph.D. Qualifying Examination

Spring 2006

Design and Analysis of Algorithms

Question I (100 points)

1. (Generating Permutations, 50 points) Provide a high-level efficient solution for generating all n! permutations of {1, …, n}.
   - Show the correctness of your solution using {1,2,3}.
   - Discuss time-space complexity of your solution.

2. (Searching, 50 points) Develop a searching strategy in the 2-D Euclidean space by walking from source S=(0,0) to an unknown location D=(x, y). However, $y = 2x + c$ for D, with a known constant c. The total moving distance during the searching process should be limited to a constant factor of the shortest path between S and D: dis(S,D).
   - Demonstrate the correctness of your searching method.
   - Briefly discuss the value of the constant factor with respect to dis(S,D).

Ph.D. Qualifying Examination

Spring 2006

Design and Analysis of Algorithms

Question II (100 points)

60 points

(1) Find an asymptotic tight bound for the recurrence:

$$T(n) = 2T(\frac{n}{2}) + 6n^2 \log_2 n$$

Justify your answer.

40 points

(2) Design an algorithm that, given n integers in the range 0 to k, preprocesses its input and then answers any query about how many of the n integers fall into a range $[a...b]$ in $O(1)$ time. Assume that a and b are integers in the range 0 to k. Your algorithm should use $\Theta(n+k)$ preprocessing time. Show the pseudocode of your algorithm and explain how the queries are solved in $O(1)$ time.

Ph.D. Qualifying Examination

Fall 2006

Design and Analysis of Algorithms

Question I (100 points)

1. (Searching 50 points) Write an $O(n)$ algorithm that receives as input an array A of n real numbers sorted in non-decreasing order and a value v. The algorithm returns true if there are distinct indexes i and j such that $A[i] + A[j] = v$ and false otherwise.

2. (Greedy Algorithms, 50 points) A private plane flying from Fort Lauderdale to JFK in New York must stop periodically to refuel. Develop a greedy algorithm to minimize the number of stops for refuel, and prove that your algorithm is optimal.

Ph.D. Qualifying Examination

Fall 2006

Algorithms

Question II (100 points)

50 points
(1) Find an asymptotic tight bound for the recurrence:

$$T(n) = 4T(\frac{n}{2}) + n^2\sqrt{n}$$

50 points
(2) Given a sorted array of distinct integers *A[1, ..., n]*, we want to find out whether there is an index *i* for which *A[i] = i*. Give the pseudo-code of a divide-and-conquer algorithm that runs in time *O(log n)*. Justify why the running time of your algorithm is *O(log n)*.

Ph.D. Qualifying Examination

Spring 2005

Algorithms

Question I (100 points)

1. (20 points) Explain the difference and similarity between dynamic programming and recursion approaches. Under what situations does dynamic programming have a significant advantage over recursion, and why?

2. (50 points) Provide pseudo code using (1) dynamic programming, and (2) recursion for the following function where n and k are natural numbers:

   $$G(n, k) = G(n, k-1) + G(n-1, k) \text{ for } n > k > 0$$

   and $G(n, k) = 1$ when $k=0$ or $k=n$.

3. (30 points) Show how $G(4,2)$ is calculated under both solutions. Provide details of the order of recursive calls in the recursive solution and the tabular method in the dynamic programming solution.

Ph.D. Qualifying Examination

Spring 2005

Algorithms

Question II (100 points)

Part 1 (50 points)

For each pair of expressions (A,B) in the table below, indicate whether A is O, o, Ω, ω, or Θ of B. Your answer should be in the form of the table, with "*yes*", "*no*", or "*undefined*" written in each box.
(You do not have to show your work for Part 1)

A	B	A=O(B)	A=o(B)	A=Ω(B)	A=ω(B)	A=Θ(B)
$100n^2 - 50n$	n^2					
$\lg^{100} n$	n^5					
2^n	n^{50}					
$n^{\lg 8}$	$5^{\lg n}$					
$\sqrt{n}$	$n^{\sin n}$					

Part 2 (50 points)

Although merge sort runs in $\Theta(n \log_2 n)$ worst-case time and insertion sort runs in $\Theta(n^2)$ worst-case time, the constant factors in insertion sort make it faster for small *n*. Thus, it makes sense to use insertion sort within merge sort when subproblems become sufficiently small. Consider a modification to merge sort in which *n/k* sublists of length *k* are sorted using insertion sort and then merged using the standard merging mechanism.

a. (20 points) Show that the *n/k* sublists, each of length *k*, can be sorted by insertion sort in $\Theta(nk)$ worst-case time.
b. (30 points) Show that the sublists can be merged in $\Theta(n \log_2(n/k))$ worst-case time.

PhD Qualifying Examination
Fall 2005
Algorithms

Question I (100 pts)

A minimum edge weighted matching corresponds to a matching of n positive points to n negative points distributed in a 2-D space. In such a matching each positive point is paired with a distinct negative point. The distance of a matching pair is measured by the Euclidean distance between two points. Consider a greedy matching algorithm working in rounds. At each round, one positive point and one negative point that are the closest in the candidate pool are selected. This matching pair is then removed from the pool.

1. Use plain English to describe an efficient implementation of the greedy matching algorithm. Specify the data structure used.
2. Analyze the complexity of your implementation in terms of n.
3. Will this greedy matching algorithm guarantee a matching with a minimum total matching distance? If yes, provide a proof; otherwise, show a counter example.

Ph.D. Qualifying Examination

Fall 2005

Algorithms

Question II (100 points)

50 points

(1) Find an asymptotic tight bound for the recurrence:

$$T(n) = 81T(\frac{n}{9}) + n^4 \log_2 n$$

Justify your answer.

50 points

(2) An array *A[1..n]* contains all integers from *0* to *n* except one. Each integer is stored as *k*-bit binary number. For example, if $k = 2$ and the array $A = [01, 00, 11]$, then the missing number is *10*. The only operation to examine integers is BIT-LOOKUP(i, j), which returns the *j*th bit of number *A[i]*. Assume that BIT-LOOKUP(i, j) takes a constant time. Design an *O(n)* algorithm to find the missing number. Justify that your algorithm has worst-case running time *O(n)*.