

Analysis of Algorithms Qualifying Exam  
Spring 2002  
February 17, 2002

All problems have the same weight (10 pts). Some are easier, some harder. You should try as many as possible, although it is NOT expected that you will be able to answer all of them in the time given.

1. Hashing:

- (a) Define a *Universal Collection of Hash Functions*. (3 pts)
- (b) Prove: if  $h$  is chosen from a universal collection of hash functions and is used to hash  $n$  keys into a table of size  $m$ , where  $n \leq m$ , the expected number of collisions involving a particular key is less than 1. (7 pts)

Hints: for keys  $x$  and  $y$ , let  $c_{xy}$  be the random variable that is 1 if  $h(x) = h(y)$  and 0 otherwise. What is the expected value of  $c_{xy}$ ? What is the expected value of the *total* number of collisions involving a key  $x$ ?

2. Worst Case Analysis:

Consider a priority queue that supports the operations **INSERT** and **EXTRACT-MIN**. Argue that in the worst case, if we perform a mixture of  $n$  **INSERT** and **EXTRACT-MIN** operations on the priority queue, there is at least one operation that takes  $\Omega(\lg(n))$  time. Assume that the order of elements in the priority queue is determined by comparisons only. (Hint: For any set of numbers, at least one number must equal or exceed the average.)

3. Solving a Recurrence :

Prove by induction that the recurrence

$$T(n) = T(n/2) + n^2, \text{ for } n \geq 2, \text{ with } T(1) = 0,$$

has solution  $T(n) = 4(n^2 - 1)/3$ . You may limit your proof to the case where  $n$  takes values that are powers of 2.

4. Sorting:

Describe *Radix Sort* and prove that *Radix Sort* sorts in linear time.

5. Optimization:

- (a) (5 pts) Describe Huffman's Algorithm for the construction of Huffman Codes. Use character-based text as the domain.
- (b) (5 pts). Is this an example of a *Greedy Algorithm* or of a *Dynamic Programming* one? You **must** give descriptions of both methods in sufficient detail to justify your claim.

6. Approximation:

Prove that, assuming the cost function satisfies the triangle inequality, there is an approximate Traveling Salesman algorithm that produces a tour in time  $\Theta(|V|^2)$ , where  $V$  is the set of vertices, and the tour produced has cost no worse than twice the cost of the optimal one.

7. Union-Find:

The notions of **rank**, **union by rank** and **path compression** are central to the **Union-Find** problem.

- (a) (3 pts) What is the **Union-Find** problem? In particular, what operations must be supported?
- (b) (2 pts) Describe **union by rank**.
- (c) (5 pts) Sketch the proof of the statement that the operations on a disjoint-set forest with union by rank but without path compression run in  $\mathbf{O}(m \lg(n))$  time, where  $m$  is the number of operations and  $n$  is the total number of items in the sets.

8. Balanced Trees:

Consider a red-black tree on  $n$  keys that maintains, in each node of the tree, the height of the node. (Recall that the height of a node is the maximum number of edges on any simple path from the node to one of its descendant leaves.) Describe how to update this height information efficiently when a rotation occurs. Analyze the running time of your update algorithm.

9. Graphs:

- (a) (5 pts) Describe Kruskal's Algorithm for the construction of a minimum spanning tree for a given weighted undirected graph  $\mathbf{G}$ .
- (b) (5 pts) Suppose all edge weights in a graph are integers in the range from 1 to  $|V|$ . How fast can you make Kruskal's algorithm run? Sketch a proof of your claim.

10. Flow Networks.

Show the execution of the Edmonds-Karp algorithm on the flow network

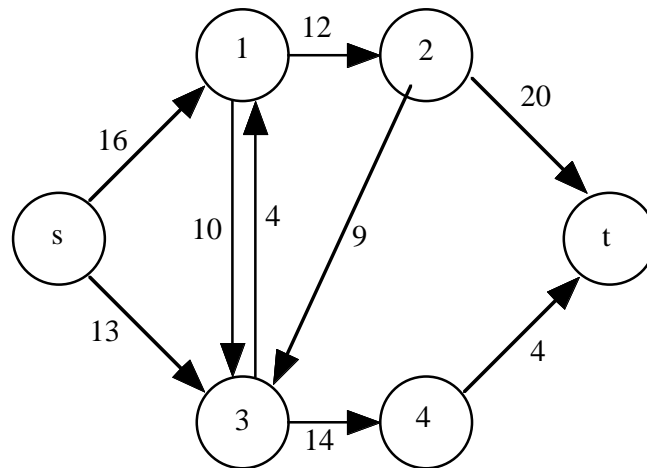


Figure 1 : Flow Network

Hint: The Edmonds-Karp algorithm uses the shortest path (= with the smallest number of edges) from  $s$  to  $t$  in the residual graph as the next path to use for the augmentation of the flow.

11. Number Theory & Cryptography:

Describe the RSA algorithm. You must give sufficient detail so that the reader should be able to implement it knowing just the elementary number theory at its foundations.

12. Matrix Multiplication:

Let  $M(n)$  be the time to multiply two  $n \times n$  matrices and let  $S(n)$  denote the time to square an  $n \times n$  matrix. Show that multiplying and squaring matrices have essentially the same difficulty. More specifically, prove that an  $M(n)$ -time multiplication algorithm implies an  $O(M(n))$ -time squaring algorithm, and an  $S(n)$ -time squaring algorithm implies an  $O(S(n))$ -time multiplication algorithm.

13. Discrete Fourier Transform:

Compute the Discrete Fourier Transform of the sequence  $[0, 1, 2, 3]$  with respect to the ring of Complex Numbers. Use the FFT algorithm and carry out the computations.

14. NP-vs-P:

- (a) (2 pts) Describe the Clique Problem.
- (b) (2 pts) Describe the Vertex-Cover Problem.
- (c) (6 pts) Starting from the NP-completeness of the Clique Problem, prove the NP-completeness of the Vertex-Cover Problem.