

Final*DUE DATE: Dec 11, 2007***Instructions:**

- The exam **MUST** be submitted no later than 5pm Tuesday Dec 11, whether electronically or by hand.
- The exam is open-book: you may use any reference material from the textbook, or from the Cormen/Leiserson/Rivest/Stein book. You **MAY NOT** use answers found on the web.
- If you need a desired result (running time of Kruskal's algorithm, etc) that is not directly related to the problem you're solving, you may cite it. For example, you can say, "We can compute an MST in $O(m \log n)$ time using Kruskal's algorithm (Kleinberg/Tardos, Chapter 4.5). Citations must come from one of the two textbooks indicated above.
- This is an examination: **consultations are not permitted.**
- **Unless clearly specified, all statements require proofs.**

1

We return to 3SAT once again for this problem. Assume we are given an instance of 3SAT in which

- There are m clauses and n variables.
- Each clause has *exactly* 3 literals
- There is a unique satisfying assignment.

We propose the following “random walk” strategy.

1. Start with a *random* assignment of values to the variables
 2. Repeat n times:
 3. If all clauses are satisfied, we are done. Else,
 4. Take any clause that remains unsatisfied, pick one of its variables with equal probability, and flip its value, thus satisfying this clause (of course, other clauses might become unsatisfied)
- a) Show that the above procedure finds a satisfying assignment with probability $(2/3)^n$
 - b) Use the above fact to find an algorithm that guarantees to find a satisfying assignment. Give a bound on its running time and specify whether this bound is expected, or worst-case.

2

The Xenopic rabbits spend their lives jumping from the place of their birth to Carrotopia, the semi-mythical land of the everlasting carrots. But these are weak rabbits; every time they jump a distance, they wear down, and can only jump half the distance the next time. Life starts with a mighty leap halfway towards Carrotopia, and progresses by halving ever after.

One enterprising youngster Monte C. feels that life is too predictable and decides to spice things up. Rather than following the boring deterministic ways of his elders, he decides to indulge in some illicit randomization.

Let's say that Carrotopia is at the center of the land: we call this point 0. The rabbits are born far away, in a small town called N^1 . Coincidentally, N happens to be exactly N miles away from Carrotopia.

It turns out that randomness works differently in different parts of the country. In fact, at x units away from Carrotopia, a random sample of the reals generates a number that has an expected value of $g(x)$. Monte starts from home, and rolls a die; he then jumps the value that comes up. He then computes a random sample from the town he's in, rolls it, and jumps again. All he can rely on is the fact that $g(x)$ is a monotonically increasing function of x (for example, the deterministic rabbits are operating with a *deterministic* function $g(x) = x/2$).

When should we expect Monte to reach Carrotopia ? Your answer should be in terms of N and $g(x)$ (don't expect a completely closed form).

¹The remaining letters washed away in a deluge. Stuff happens.

3

3.1

An instance of KNAPSACK consists of n items, each with weight w_i and profit p_i . The goal is to maximize the total profit of items thrown into a sack that can carry a total load of at most 1.

KNAPSACK is NP-Complete. Consider however the following *fractional* variant: we're allowed to pick an α_i fraction of item i , and thus the new goal is to maximize the quantity $\sum \alpha_i p_i$, subject to the constraint $\sum \alpha_i w_i \leq 1$. Give a strongly polynomial time algorithm that finds the optimal fractional knapsack solution. Your solution will consist of the fractions α_i (note that any item not picked merely has $\alpha_i = 0$).

4

We say that a sequence of numbers x_1, x_2, \dots, x_n is *oscillating* if for any odd index i , $x_i \geq x_{i+1}$, and for any even index i , $x_i \leq x_{i+1}$. For example, the sequence $\{2, 1, 7, 3, 10, 5, 8\}$ is oscillating.

Design an efficient algorithm to find a longest oscillating subsequence of an input sequence of numbers.

5

We've seen how to multiply two n -bit integers in time $O(n^{\log 3})$. Assume now that we have a computer with a word length of \sqrt{n} , which means that we can multiply two \sqrt{n} -bit integers in constant time.

Under this cost model, show that two n -bit integers can be multiplied in time $o(n^{1.3})$.

6

We revisit the problem of computing a maximum matching in a bipartite graph $G = (L \cup R, E)$, where $|L \cup R| = n, |E| = m$. In the max-flow formulation we studied earlier, an augmenting path from the source to sink consisted of one edge from the source, a series of edges originating at an unmatched node in L and ending at another unmatched node in R , and finally an edge to t .

Let's ignore the edges from s and to t . What we have left is a path that alternates between edges not in the current matching and edges that are. The importance of such a path is this: it always has odd length (say $2k + 1$), and of these edges, $k + 1$ are not in the matching, and k are. This means that given the current matching M and such an augmenting path P , the new set of edges $M' = M \Delta P$ (where $A \Delta B = (A - B) \cup (B - A)$ is the symmetric difference between sets A and B) is of size $|M'| = |M| + 1$.

Thus, repeatedly finding such augmenting paths and computing the resulting symmetric difference allows us to continually increase the size of the matching till we reach the maximum.

6.1

Let M and M' be two matchings, where $|M| \leq |M'|$. Show that in the graph induced by the edges of $M \Delta M'$, each vertex has degree at most two. Prove that this implies that this graph consists of cycles and paths, and show that it contains at least $|M'| - |M|$ vertex-disjoint augmenting paths with respect to M .

6.2

Suppose we only consider augmenting paths of length 1 (i.e each augmenting path consists of a single edge between two unmatched nodes). Show that the matching we obtain by augmenting by **all** such paths is a $(1/2)$ -approximation to the maximum matching.

6.3

Suppose we now consider augmenting paths of length at most $2\ell + 1$. Generalize the above result to show that the best matching thus obtained is an $(\ell/(\ell + 1))$ -approximation to the maximum matching.

6.4

Assume the existence of two procedures:

1. A procedure that in time $O(m)$ produces an augmenting path with respect to the current matching.
2. A procedure that takes a parameter l , and in time $O(m)$ returns a maximal vertex-disjoint set of augmenting paths of length l . You can further assume (it's actually true!) that there are no other augmenting path of length l .

Consider the following algorithm:

1. Set $M = \emptyset$
2. For $i = 1 \dots \sqrt{n}$
3. Find a maximal vertex-disjoint set of augmenting paths of length i , and augment M with all these paths.
4. While any augmenting path remains
5. Augment M with this path

It's clear that the `for` loop runs in time $O(m\sqrt{n})$. Show that after the loop completes, there are at most \sqrt{n} augmenting paths remaining (with arbitrary lengths), and therefore the entire algorithm runs in time $O(m\sqrt{n})$.