

15-750 — Graduate Algorithms — Spring 2008

Miller and Sinop and Wu

Assignment 0 Due date: Friday, January 25

Some Reminders:

- Read the Policies section on the course web site before you start working on this assignment. Collaboration is not permitted for this assignment.
- You should refrain from using outside sources when solving these problems. For each problem, state whether you have seen it before. If you have questions, contact the course staff.
- We prefer that you type up your solutions (preferably using LaTeX). You may neatly hand-write your solutions, but if we have trouble reading them you will be required to type up future solutions.

1 Asymptotic Notation

[10 points] For each list of functions, order them according to increasing asymptotic growth. Provide a brief argument justifying each successive step in the ordering. Your answers should look something like: $n^{1/2} < n = 5n < n^2$. Don't forget to mention if $f(n) = \Theta(g(n))$. **Disclaimer:** it is *not* standard notation to write $n^{1/2} < n = 5n < n^2$ under big-O notation. Nor is it standard to write $\Theta(n^{1/2}) = o(n) = \Theta(5n) = o(n^2)$. When writing a paper, it would be better to say “ $n^{1/2} = o(n)$ and $n = \Theta(5n)$ and $5n = o(n^2)$,” but this would be longer for you to write and for us to grade.

List 1, fast growing functions: $(\log^* n)^{\log n}, n^{\log^* n}, n!, 2^{n\sqrt{\log n}}$.

List 2, slow growing functions: $2^{\log^* n}, 2^{\sqrt{\log n}}, \log \log \sqrt{n}, n, \sqrt{n}$

Recall that $\log^*(n)$ (the “log star” function) calculates how many times you would need to take the iterated \log_2 of n before you would go below 2. Formally, $\log^*(x) = 0$ for all $x \in (0, 2)$, and for all $x \geq 2$, $\log^*(x) = 1 + \log^*(\log_2(x))$. Thus for example $\log^*(2) = 1$, $\log^*(2^2) = 2$, and $\log^*(2^{(2^2)}) = 3$.

2 Whack-a-Mole

[10 points] In the carnival game of *Whack a mole*, there are n moles numbered $1, 2, \dots, n$ all in a line. Each mole sits in a hole. Each second one of the moles sticks its head out of its hole. You then have 1 second to whack the mole with a (cartoonish) mallet. If you are successful you get some points.

Let's say you know in advance the sequence of moles that will be popping up. So at time t ($1 \leq t \leq T$) let m_t denote the mole that will pop up at that time. If you successfully whack that mole you get p_t points. Also, the mallet is heavy, so that in 1 second you can only move the mallet

a distance of 1. Thus the sequence of choices of what to whack must have the property that two whacks that are k apart must occur at times that differ by at least k . You are allowed to pick any desired starting point for the mallet.

Give an algorithm to efficiently compute the optimal sequence of moles to hit. The output should be a trajectory for the mallet – that is, an array indicating the position x_t of the mallet at time t . If $x_t = m_t$ then this means that we hit the mole at time t , and get p_t points. Otherwise we get 0 points at time t .

Write-up your algorithm formally in pseudo-code. Try to make your algorithm as efficient as possible. Its running time should be polynomial in n and T .

3 Upper and Lower Bounds

[10 points] Let A and B be two sorted arrays with n elements in each. We would like to find the median element in $A \cup B$ (n th smallest element).

- (a) Give an efficient algorithm for finding the median of $A \cup B$ given A and B .
- (b) Give a lower bound on running time by showing that any comparison-based algorithm must make $\Omega(f(n))$ comparisons at the worst case.

4 Probability

[10 points] For a given graph $G = (V, E)$, MAX-CUT problem is to partition vertices V into two disjoint sets, S and \bar{S} , so as to maximize the total number of edges in this cut (edges with one endpoint in S and one endpoint in \bar{S}). In general, computing MAX-CUT of an arbitrary graph is NP-hard. However we will give a $\frac{1}{2}$ -approximation algorithm for this problem (given any graph with maximum cut value c , the algorithm finds a cut of value at least $\frac{1}{2}c$). Let $m = |E|$ denote the number of edges in this graph.

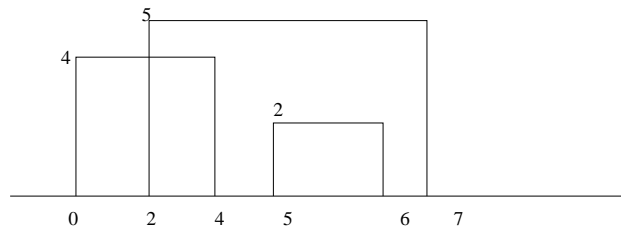
Consider flipping an unbiased coin for each vertex $v \in V$, and place vertex v in S if it is head, or in \bar{S} if it is tails. So $P(v \in S) = P(v \in \bar{S}) = \frac{1}{2}$.

- (a) For any given edge $e_{ij} \in E$ between vertices i and j , what is the probability that this edge is cut by (S, \bar{S}) ?
- (b) What is the expected value of this cut? (Use indicator variables for determining whether if an edge is cut, and then use linearity of expectation).
- (c) Give a deterministic algorithm which also achieves the same approximation ratio (consider locally improving the given partition and show that it has the same approximation ratio).

5 The Silhouette

[10 points] Suppose that we are given a set of n 2D rectangular buildings. Each building is sitting on the ground at height zero and it is given by three real numbers $B_i = (l_i, r_i, h_i)$ where l_i is the left side, r_i is the right, and h_i is the height of the building. The goal is to compute the silhouette

of the set of buildings. You should return a rectilinear path from left to right (see Figure 1). Your algorithm should run in $O(n \log n)$ time.



Input: $l1 = 0, r1 = 4, h1 = 4$
 $l2 = 2, r2 = 7, h2 = 5$
 $l3 = 5, r3 = 6, h3 = 2$

Output: $(0,0) - (0,4) - (2,4) - (2,5) - (7,5) - (7,0)$

Figure 1: An example input and expected output.