Intro to Algorithms, COMP-160, Homework #5

Benjamin Tanen, 03/03/2016

2. You are given k coins, arranged in two stacks. A **move** involves choosing a stack and redistributing its coins into other stacks, but when doing this you can place at most one coin in each other stack. During the move, you are allowed to make as many new stacks as you like, but only one coin can go in each. The *reward* of one move is precisely the number of coins in the stack that you choose to redistribute. Suppose that you get to make some huge number of moves (say, n).

   (a) Given any number of coins k in any orientation, with a certain constant amount of work, we can orient our k coins into a perfect triangle (size of base b = size of height h).

   Note: For some values of k there will be additional left over coins. However, once we obtain the perfect triangle plus these additional garbage coins, we can continue to ignore these extra coins because we will continue to maintain a perfect triangle, given the following moves.

   Once we have this perfect triangle (potentially with some additional single coins), we can maximize our reward for each round by choosing the largest stack (the stack with height h) and redistributing it to the rest of our stacks. Since we had h stacks, redistributing these h coins will again result in a perfect triangle (adding one to the other $h-1$ existing stacks and making a new stack of one coin). Since our perfect triangle is maintained, at this point we can get a reward of h each time.

   Since we know we can get a reward of h each time, we should find a value of h in terms of our known value k. We can describe k as the sum of the coins in each stack. Once we have a perfect triangle (where there are h stacks with one to h coins), we can see that:

$$k = \sum_{i=1}^{h} s_i = \frac{h(h+1)}{2} = \frac{h^2}{2} + \frac{h}{2}$$

   If we then attempt to isolate h, we can get $h = \sqrt{(}2k-h)$. Since $h \leq k$ (a single stack has a maximum value of k), we know that $\sqrt{k} \leq \sqrt{(}2k-h)$, which leads us to say that $\sqrt{k}$ is an under-approximation of h. Therefore we can see that $h = f(k) = \Omega(\sqrt{k})$, so our average reward has a lower bound of $\sqrt{k}$.

   (b) For this problem, we describe the potential after any move i to be ϕ_i. Say we describe the potential to be the number of coins above our expected average cost $\sqrt{k}$, so $\phi_i = \sum_{i=1}(s_i - \sqrt{k})_+$ (**Note:** the $_+$ denotes using the plus-function so $\phi_i \geq 0$)

   From here, we can define any stack in our game as either *large* ($h \geq \sqrt{k}$) or *small* ($h < \sqrt{k}$). We now want to define $\hat{c_i} = c_i + \Delta\phi_i$ for either type of stack.

Consider the case where we move a *small* stack. Our actual cost is $c_i < \sqrt{k}$ (the height of our stack). Next we want to consider how much redistributing this stack will change the potential, or how many coins will be redistributed to stacks of height $h \geq \sqrt{k}$. Since we can never have more than $\sqrt{k}$ stacks of height $h \geq \sqrt{k}$ (constrained by us only having k coins), we know that we can only add one coin to a maximum of $\sqrt{k}$ stacks. This means that $\Delta\phi_i \leq \sqrt{k}$ for the redistribution of any *small* stack. Therefore, we get that $\hat{c_i} = c_i + \Delta\phi_i < \sqrt{k} + \sqrt{k} = 2\sqrt{k}$. Thus, we can see that in the case of a *small* stack, $\hat{c_i} = O(\sqrt{k})$.

Next, consider the case where we move a *large* stack. Our actual cost is $c_i \geq \sqrt{k}$, but we can more specifically define $c_i = \sqrt{k} + x$ where $x \geq 0$. Next we want to consider how many coins will be redistributed to stacks of height $h \geq \sqrt{k}$. Again, since we can never have more than $\sqrt{k}$ stacks of height $h \geq \sqrt{k}$, we can only add one coin to a maximum of $\sqrt{k}$ stacks. The remaining x coins will be redistributed to stacks that are below $\sqrt{k}$ (or by making new stacks), so this decreases our potential by the number of coins x redistributed in this way, so we get $\Delta\phi_i \leq \sqrt{k} - x$. Therefore, $\hat{c_i} = c_i + \Delta\phi_i \leq \sqrt{k} + x + \sqrt{k} - x = 2\sqrt{k}$. Thus, we can see in the case of a *large* stack, $\hat{c_i} = O(\sqrt{k})$.

Since we can see that choosing any stack (either *large* or *small*), we get $\hat{c_i} = O(\sqrt{k})$. To calculate our average cost per move, we can see:

$$\frac{\sum_{i=1}^{n} \hat{c_i}}{n} = \frac{\sum_{i=1}^{n} O(\sqrt{k})}{n} = \frac{n \cdot O(\sqrt{k})}{n} = O(\sqrt{k})$$

This shows us that the best upper bound on our reward (or cost) is $\sqrt{k}$, or that $f(k) = O(\sqrt{k})$.