Cpt S 450 Homework #4 Solutions

1. First, we need basics in Probability Theory: Assume that we have n coins to toss where, for each coind, it has probability p of falling into a bag and probability $(1-p)$ of falling out of the bag. We use r to denote the number of coins in the bag after all the n coins tossed. Then, this random variable has this distribution: $Prob(r) = C_n^r \cdot p^r \cdot (1-p)^{n-r}$. This is called binomial distributionderived from Bernolli formula. In this hw, we take $p = .5$. Notice also that, when n gets big, the distribution can be approximated by a normal distribution with mean $\mu = \frac{n}{2}$ and standard deviation $\sigma = \frac{\sqrt{n}}{2}$. In particular, $Prob(\mu)$ can be approximated by the density of the normal distribution at the mean which is $\frac{1}{\sigma\sqrt{2\pi}}$, and $Prob(\mu-\sigma) = Prob(\mu+\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}}$.

There is a very important property about normal distributions. The probability of r staying close to the mean μ is very high. That is,

$$\sum_{\mu-\sigma\leq r\leq\mu+\sigma} Prob(r) \geq \frac{2}{3}. \tag{1}$$

Now, we are ready to start with the solution.

Step1. Write a formula. We use $T(n)$ to denote the average case complexity that we are looking for. Clearly,

$$T(n) = \sum_{0\leq r\leq n} Prob(r) \cdot (T(r) + T(n-r) + O(r(n-r))).$$

Step2. Guess a solution: $T(n) = O(n^2 \log n) = cn^2 \log n$.

Step3. Check the solution.

We set $X(r) := T(r) + T(n-r) + O(r(n-r))$.

We divide all the r's into two sets: $B_2 = \{r : \mu - \sigma \leq r \leq \mu + \sigma\}$ and B_1 is the rest of r's. Clearly,

$$T(n) = \sum_{r\in B_1} Prob(r) \cdot X(r) + \sum_{r\in B_2} Prob(r) \cdot X(r).$$

First, notice that $X(r)$ grows at least quadratic in r (on the left half of B_1); hence we can observe the following:

$$X(\mu-\sigma) \geq 9 \cdot X(\frac{1}{6} \cdot (\mu - \sigma)).$$

From (1) and the fact that X is convex, we have,

$$T(n) \leq 2 \cdot X(\frac{1}{6} \cdot (\mu - \sigma)) + \sum_{r\in B_2} Prob(r) \cdot X(r). \tag{2}$$

Notice that, $X(\mu-\sigma) \leq X(r)$ for every $r \in B_2$. It is not hard to how that

$$X(\frac{1}{6}\cdot(\mu-\sigma)) \leq \frac{1}{6}\cdot\frac{2}{3}\cdot X(\mu-\sigma)$$

$$\leq \frac{1}{6}\cdot\sum_{r\in B_2} Prob(r)\cdot X(r).$$

Therefore, (2) can be written into:

$$T(n) \leq (1+\frac{1}{3})\cdot\sum_{r\in B_2} Prob(r)\cdot X(r). \tag{3}$$

From (3) and the fact that $Prob(r) \leq Prob(\mu)$ for all r, we have,

$$T(n) \leq (1+\frac{1}{3})\cdot\frac{1}{\sigma\sqrt{2\pi}}\sum_{r\in B_2} X(r). \tag{4}$$

That is,

$$T(n) \leq (1+\frac{1}{3})\cdot\frac{1}{\sigma\sqrt{2\pi}}\sum_{r\in B_2} T(r)+T(n-r)+O(r(n-r)). \tag{5}$$

To make a little simplification, we obtain:

$$T(n) \leq (1+\frac{1}{3})\cdot\frac{1}{\sigma\sqrt{2\pi}}\cdot(2\cdot\sum_{r\in B_2} T(r)+\sum_{r\in B_2} O(r(n-r))). \tag{6}$$

One can easily compute the $\sum_{r\in B_2} O(r(n-r))$ which is $O(n^{2.5})$ by taking integral etc. (Note: I always have a secret weapon in hand: to compute $f(x+\delta)-f(x-\delta)$, I use the approximation $f'(x)\cdot 2\delta$, when δ is small.) So, now, (6) is changed into:

$$T(n) \leq (1+\frac{1}{3})\cdot\frac{1}{\sigma\sqrt{2\pi}}\cdot 2\cdot\sum_{r\in B_2} T(r)+\frac{O(n^{2.5})}{\sigma}. \tag{7}$$

Recalling the definition of σ, we have

$$T(n) \leq (1+\frac{1}{3})\cdot\frac{1}{\sigma\sqrt{2\pi}}\cdot 2\cdot\sum_{r\in B_2} T(r)+O(n^2). \tag{8}$$

Now, we really start the checking step. Plug-in the assumption that $T(r) \le cr^2 \log r$ and the definition of B_2, we have

$$\sum_{r \in B_2} T(r) \le \sum_{\mu-\sigma \le r \le \mu+\sigma} cr^2 \log r. \tag{9}$$

The right hand side can be easily calculated using intergral and my secret weapon mentioned earlier: $\sum_{\mu-\sigma \le r \le \mu+\sigma} cr^2 \log r$ can be arrpoximated by

$$c \cdot (\frac{n}{2})^3 \cdot \log \frac{n}{2} \cdot \frac{3 \cdot 2}{\sqrt{n}} \tag{10}$$

Hence, (8) can now be written into:

$$T(n) \le (1 + \frac{1}{3}) \cdot \frac{1}{\sigma\sqrt{2\pi}} \cdot 2 \cdot c \cdot (\frac{n}{2})^3 \cdot \log \frac{n}{2} \cdot \frac{3 \cdot 2}{\sqrt{n}} + O(n^2). \tag{11}$$

Using the definition of σ over (11) and one can easily show that, directly from (11),

$$T(n) \le c \cdot n^2 \cdot \log n. \tag{12}$$

Hence, our guess is checked.

Finally, the average time complexity is $O(n^2 \log n)$.

2. When $i = 2$, the step in the for-loop takes time $O(n^{1.59})$. When $i = 3$, the step incolves the product of a $2n$-length bit string and a n-length bit string, which takes time $O((2n)^{1.59})$. When $i = 4$, the step incolves the product of a $3n$-length bit string and a n-length bit string, which takes time $O((3n)^{1.59})$, and so on. Hence, in worst cases, the naiveKaratsuba takes time,

$\sum_{k=1}^{n-1} O((kn)^{1.59})$
$= O(n^{1.59}) \sum_{k=1}^{n-1} k^{1.59}$ (use integral)
$= O(n^{1.59} \cdot n^{2.59})$
$= O(n^{4.18})$.

3. We use $BK(m, n)$ to denote the worst-case complexity of betterKaratsuba running on m number of n-bit strings. Clearly, the desired worst-case complexity in the problem is to compute $BK(n, n)$.

Notice that $BK(n, n)$ should be bounded by the summation of

- the complexity of betterKaratsuba over the first group; i.e., $BK(\frac{n}{2}, n)$;
- the complexity of betterKaratsuba over the second group; i.e., $BK(\frac{n}{2}, n)$;

- the complexity of Karatsuba over the results of the first and the second groups – noticing that each result in worst cases is a $\frac{n}{2} \cdot n$-length bit string. Hence, the compelxity is $O((\frac{n}{2} \cdot n)^{1.59})$.

To sump up, $BK(n,n) \leq 2 \cdot BK(\frac{n}{2}, n) + O((\frac{n}{2} \cdot n)^{1.59})$. In above, let us put the constant in the O back, say some b. That is,

$BK(n,n) \leq 2 \cdot BK(\frac{n}{2}, n) + b \cdot (\frac{n}{2} \cdot n)^{1.59}$.

It is hard to guess a solution now. But let us transform $BK(\cdot, n)$ into $G(\cdot)$ (i.e., try to ignore the second parameter – this is because it does not change in both sides of the inequality above):

$$G(n) \leq 2 \cdot G(\frac{n}{2}) + b \cdot (\frac{n}{2} \cdot n)^{1.59}.$$

Now, we guess the solution for $G(n) = O((n^2)^{1.59})$. Let $G(n) = c \cdot (n^2)^{1.59}$. We have,

$LHS \leq$
$2 \cdot c \cdot (\frac{n}{2})^{2 \cdot 1.59} + b \cdot (\frac{n}{2} \cdot n)^{1.59} \leq$
(you need to verify, by taking c large)
$c \cdot (n^2)^{1.59}$.

Hence, the worst case compelxity of betterKaratsuba runs in $O((n^2)^{1.59})$ (i.e., $O(n^{3.18})$).

4. Each cube, shown in Figure 1, in a 3-dimention space can be specified by a pair (d, l); the corner point d and its size l

The corner point d is specified by its three corordinates (x, y, z).

For the problem set-up, one can assume, without loss of generality, that all the airplanes are located in the cube whose corner point is the origina and whose size is n, we call this cube as the entire space S.

A cube is a *unit-cube* if

- the size $l = 1$,
- each of x, y, z is an integer in $\{0, 1, \cdots, n\}$,
- the cube is contained in the entire space S.

A cube is a *half-cube* if

- the size $l = .5$,
- each of x, y, z is either an integer k, or $k+.5$, for some $k \in \{0, 1, \cdots, n\}$.

A cube is a *eight-cube* if

- the size $l = 8$,
- each of x, y, z is an integer in $\{0, 1, \cdots, n\}$.

There are exactly n^3 unit-cubes and there are $O(n^3)$ eight-cubes. Since there are only n^3 airplanes, we have two cases to consider:

- there is exactly one airplane in each of the n^3 unit-cubes. In this case, the closest pair can not stay farther than $2 \cdot \sqrt{3} < 4$.
- there are at least two airplanes in some unit-cube. In this case, the closest pair can not stay farther than $\sqrt{3} < 4$.

Hence, the closest pair can not stay farther than 4. This says that the closest pair can be found in one of the $O(n^3)$ eight-cubes. How many airplanes in any eight-cube? Notice that there is at most one airplane in a half-cube. Hence, there are at most 16^3 airplanes in any eight-cube. Hence, identifying the closest pair in a eight-cube takes constant time. Since there are at most $O(n^3)$ eight-cubes, the total time is bounded by $O(n^3)$.

5. For each string α, one first compute a point $(\#_a(\alpha), \#_b(\alpha))$ in the two-dimention plane. Hence, from the array A of n strings, one come up with n points. One can run the closestpair algorithm on the n points – this will give you the closest distance *dist* between two points in the n points. Then, the smallest difference of all the distinct pairs in A is simply $dist^2$. Up to now, the total time is the sum of $O(nm)$ (to compute the points for each string in A) and $O(n \log n)$ (the time in running the closestpair algorithm).

Notice that, when $m < \log n$, there must be some $i \neq j$ satisfying $A[i] = A[j]$ (i.e., there are repeated elements in A) and this will make the smallest difference of all the distinct pairs in A be 0. (no closestpair algorithm needs to be run; i.e., the term $O(n \log n)$ need not be included in the sum)

Hence, the total running time is bounded by $O(nm)$.