CS 30A : Discussion

Week 2 : Homework 1 Solutions

Zexi Huang

Outlines:

1. Inductive Proof for Complexity
   - Problem #9
2. Algorithm Design and Analysis
   - Problem #10
   - Problem #11

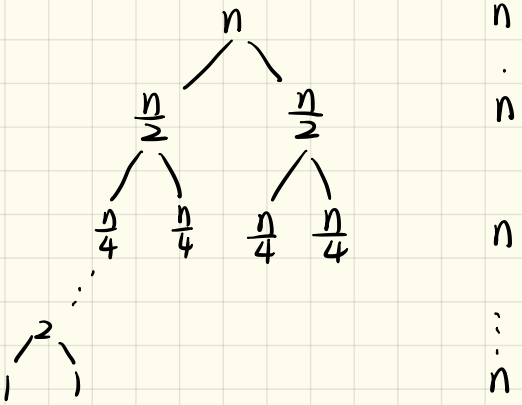
9. (10 points) Prove by induction that T(n) is $\Theta(n \log n)$ for the following recurrence:

$T(1) = 1$

$T(n) = 2T(\lfloor n/2 \rfloor) + n$

You can use the recurrence tree to understand the structure of the recursion, but the proof has to be by induction. Be sure to write down the induction hypothesis and verify the base case.

(1) Recurrence Tree



$\Theta(n \log_2 n)$

(2) $T(n) = \Theta(n \log n) \Leftrightarrow T(n) = O(n \log n),\ T(n) = \Omega(n \log n)$

① $T(n) = O(n \log n) \Leftrightarrow T(n) \le c n \log n + d,\ \forall n \in N^*$

Induction:

(1) Base case: $T(1) = 1 \le c \cdot 1 \cdot \log 1 + d = d \Rightarrow d \ge 1$

(2) Induction Hypothesis:

$\forall n \in \{1, 2, \cdots, k-1\},\ T(n) \le c n \log n + d$

(3) Induction: $T(k) = 2T(\lfloor \frac{k}{2} \rfloor) + k$

$\le 2c \lfloor \frac{k}{2} \rfloor \log \lfloor \frac{k}{2} \rfloor + 2d + k$

$\vdots \quad \Leftarrow$

$\le c k \log k + d$

9. (10 points) Prove by induction that T(n) is $\Theta(n \log n)$ for the following recurrence:

$T(1) = 1$

$T(n) = 2T(\lfloor n/2 \rfloor) + n$

You can use the recurrence tree to understand the structure of the recursion, but the proof has to be by induction. Be sure to write down the induction hypothesis and verify the base case.

$T(k) \le 2c\lfloor \frac{k}{2} \rfloor \log \lfloor \frac{k}{2} \rfloor + 2d + k$

$\le 2c \cdot \frac{k}{2} \cdot \log \frac{k}{2} + 2d + k \qquad \lfloor x \rfloor \le x$

$= ck(\log k - \log_2 2) + 2d + k$

$= ck\log k + (1-c)k + 2d \quad \Rightarrow (1-c)k + 2d \le d \Rightarrow c \ge \frac{d}{k} + 1$

$\le ck\log k + d$

The induction is good $< \begin{matrix} d \ge 1 \\ c \ge \frac{d}{k} + 1 \Rightarrow c \ge \frac{d}{1} + 1 \ge \frac{d}{k} + 1 \end{matrix} \Rightarrow \begin{cases} d \ge 1 \\ c \ge d + 1 \end{cases}$ $\begin{matrix} d = 1 \\ c = 2 \end{matrix}$

$T(n) = O(n \log n)$

9. (10 points) Prove by induction that T(n) is $\Theta(n \log n)$ for the following recurrence:

$T(1) = 1$

$T(n) = 2T(\lfloor n/2 \rfloor) + n$

You can use the recurrence tree to understand the structure of the recursion, but the proof has to be by induction. Be sure to write down the induction hypothesis and verify the base case.

② $T(n) = \Omega(n \log n) \Leftrightarrow T(n) \geq pn\log n + q$, $\forall n$

(1) Base case: $T(1) = 1 \geq pn\log n + q = q \Rightarrow q \leq 1$

(2) Hypothesis: $\forall n \in 1 \sim k-1$.

(3) Induction: $T(k) = 2T(\lfloor \frac{k}{2} \rfloor) + k$

$\overset{H}{\geq} 2p\lfloor \frac{k}{2} \rfloor \log \lfloor \frac{k}{2} \rfloor + 2q + k$

$x-1 < \lfloor x \rfloor \leq x$

$> 2p(\frac{k}{2} - 1)\log(\frac{k}{2} - 1) + 2q + k$

$= p(k-2)(\log(k-2) - 1) + 2q + k$

$= pk\log(k-2) - 2p\log(k-2) - p(k-2) + 2q + k$ (*)

$\geq pk\log k + q$ (**)

(*) - (**)

$= p[(k-2)(\log(k-2) - 1) - k\log k)]$

$+ q + k \geq 0$

$p \leq \dfrac{q+k}{k\log k - (k-2)(\log(k-2) - 1)}$

$p \leq \min_k \left\{ \dfrac{q+k}{k\log k - (k-2)(\log(k-2) - 1)} \right\}$

Satisfying p, q: $\begin{cases} p = \frac{1}{2} \\ q = 0 \end{cases}$

(10 points) We are given a set K of n red/green points in 1D space and another set U of m points whose colors are not known. Suppose we use a scheme in which the color of an unlabeled point in U is obtained by the color of the nearest point in K. Describe an algorithm for labeling the points in U. What is the big theta time complexity of your proposed algorithm? For an extra 10 credit points, find an algorithm with a time complexity of O((m+n) log (m+n)).

Problem:

$K = \{0.9, 0.3, 2.1, -2\}$

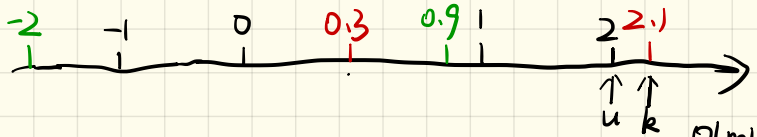
$U = \{2, 1, 0, -1\}$

① Brute Force Algorithm

```
for point u in U:                    O(m)
    for point k in K:                O(n)
        compute d(u,k)               O(1)
        update (kmin, dmin)          O(1)
    color u with kmin                O(1)
```

$O(mn+m) = O(mn)$

② Improve Brute Force Algorithm




```
Sort U, K in ascending order         O(mlogm) + O(nlogn)
for point u in U:                    O(m)
    while k<u:                       up to O(n)
        move k to right              (k = k.next) (i_k += 1)   O(n)
    compare d(u,k) and d(u,k-left), color u.   O(1)
```

$O(m\log m) + O(n\log n) + O(m) + O(n) =$

$O((m+n)\log(m+n))$

11. (10 points) We are given a sequence A[1..n] of positive numbers. We want to approximate the sequence by another sequence B[1..n] in which the values are 0 at either end and a contiguous sequence of k (to be decided) repeating values, each of magnitude m (to be decided). The cost of a specific choice of B is $\text{cost}(B) = \sum_{i=1}^{n}(A(i) - B(i))^2$. Find an algorithm for computing the sequence B (defined by the starting point, ending point, and the constant repeating value m) that minimizes the cost. For an extra 10 credit points, find an algorithm with a time complexity of $O(n^2)$.

Problem:

$A = [5, 10, 10, 3, 1]$

$B = [0, m, m, 0, 0]$

① Observation.

$$\text{cost}(B) = \sum_{i=1}^{n}(A(i) - B(i)) = \sum_{i=1}^{p-1} A^2(i) + \sum_{i=r+1}^{n} A^2(i) + \sum_{i=p}^{r}(A(i) - m)^2 \quad (1)$$

For fixed p, r, $\frac{d\,\text{cost}(B)}{dm} = 2\sum_{i=p}^{r}(m - A(i)) = 0 \Rightarrow m = \frac{\sum_{i=p}^{r} A(i)}{r-p+1}$ (2)

$\frac{d^2\,\text{cost}(B)}{dm^2} > 0$

② Brute Force Algorithm

for all (p, r) pairs: $n-1+n-2+\cdots+1 = \frac{n(n-1)}{2} = O(n^2)$

compute m with (2) — up to $O(n)$

compute cost(B) with (1) — $O(n)$

update the minimum cost — $O(1)$

return (p, r) that has minimum cost — $O(1)$

$\Downarrow$

$O(n^3)$

11. (10 points) We are given a sequence A[1..n] of positive numbers. We want to approximate the sequence by another sequence B[1..n] in which the values are 0 at either end and a contiguous sequence of k (to be decided) repeating values, each of magnitude m (to be decided). The cost of a specific choice of B is $\text{cost}(B) = \sum_{i=1}^{n}(A(i) - B(i))^2$. Find an algorithm for computing the sequence B (defined by the starting point, ending point, and the constant repeating value m) that minimizes the cost. For an extra 10 credit points, find an algorithm with a time complexity of $O(n^2)$.

③ Improve Brute Force Algorithm

for all (p, r) pairs: $O(n^2)$

~~compute m with (2)~~ Compute s', m' with (3) $O(1)$ ~~up to O(n)~~

~~compute cost(B) with (1)~~ Compute cost(B') with (4) ~~O(n)~~ $O(1)$

update the minimum cost $O(1)$

return (p, r) that has minimum cost $O(1)$

$O(n^2)$

$$m = \frac{\sum_{i=p}^{r} A(i)}{r-p+1} \quad (2)$$

$$\text{cost}(B) = \sum_{i=1}^{p-1} A^2(i) + \sum_{i=r+1}^{n} A^2(i) + \sum_{i=p}^{r}(A(i)-m)^2 \quad (1)$$