CS 4102 Written HW2 - Greedy

Eric Xie

September 16, 2016

1. Master Theorem:
$T(n) = 2T\left(\frac{n}{2}\right) + n^2$ a=2, b=2, k=2
$a < b^k$, thus this is case 1 of the Master Theorem: $T(n) \in \Theta(n^2)$

Substitution Method:
Assume $T(n) \in \Theta(n^2)$, thus $T(n) \leq cn^2$
Base Case: $n_0 = 2$
$T(2) = 2T(1) + 4 < 4c$
$2 + 4 < 4c$
$6 < 4c$
$c > \frac{3}{2}$

IH: $T(k-1) \leq c(k-1)^2$
IS: $T(k-1+1) \leq ck^2$
$2T(k/2) + k^2 \leq ck^2$
$2c(k/2)^2 + k^2 \leq ck^2$
$2c\frac{k^2}{4} + k^2 \leq ck^2$
$\frac{2c}{4} + 1 \leq c$
$2c + 4 \leq 4c$
$4 \leq 2c$
$2 \leq c$, which agrees with the assumption that $c \geq 3/2$
Thus $T(n) \in \Theta(n^2)$

Directly Solve:
$T(n) = 2T\left(\frac{n}{2}\right) + n^2$
$= 2[2T(n/4) + (n/2)^2] + n^2$
$= 4T(n/4) + \frac{n^2}{2} + n^2$
$= 2^kT(\frac{n}{2^k}) + \frac{(2^k-1)n^2}{2^{k-1}}$

$T(1) = 1$

$\frac{n}{2^k} = 1$
$n = 2^k$
$k = log_2 n$

$$T(n) = 2^{\log_2 n} T(\frac{n}{2^{\log_2 n}}) + \frac{(2^{\log_2 n} - 1)n^2}{2^{\log_2 n - 1}}$$
$= nT(\frac{n}{n}) + \frac{(n-1)n^2}{\frac{n}{2}}$
$= n + 2(n^2 - n)$
$= n + 2n^2 - 2n$
$= 2n^2 - n$
$= \Theta(n^2)$

2.
1) Base case: n=2. Do f(first index,second index) and return the index which is greater. Otherwise...
2) If n is even, do f(first half, second half), and recurse on the half that is greater
3) If n is odd, omit the median value, and do f(first half, second half)
If the result is +1 or -1, recurse on the side that is greater
Otherwise if the result is 0, return the median index
$T(n) = T(n/2) + n/2$
According to Master Theorem, a=1, b=2, k=1
Since $a < b^k$, this is case 1, which means the runtime is $\Theta(n)$

3.

```
public int median(int h1, int h2, int s1, int s2){
        int n1=h2-h1+1;
        int n2=s2-s1+1;
        int k1=ceiling((h1+h2)/2)
        int k2=ceiling((s1+s2)/2)
        int m1=happy.query(k1);
        int m2=sad.query(k2);
        if((n1==2 && n2<=2) || (n1<=2 && n2==2)
                arr=happy.subArray(h1,h2).append(sad.subArray(s1,s2))
                return arr[arr.size-2]
        if(n1==1 && n2==1)
                return max(m1,m2);
        else:
                if(m1>m2)
```

```
                return median(h1,k1,k2,s2)
        if(m1<m2)
                return median(k1,h2,s1,k2)
}
```

4.
T(n)=2T(n/2)+2n
By the Master Theorem, a=2, b=2, k=1
Since $a < b^k$, this is case 1 of the Master Theorem, meaning $T(n) \in \Theta(nlog(n))$

5.
Proof by induction:
Base Case: n=1, the algorithm selects the higher of the salaries, which is correct
IH: For n=k, the algorithm does not eliminate the true median
IS: For n=k+1, the algorithm does not eliminate the true median
F.S.C, assume that the algorithm eliminates the 1/4 of the salaries containing the median
Let the two medians be m1 and m2 such that m1¡m2

Case 1: The median is contained in the eliminated interval [1,m1]
The intervals [m1,n] and [m2,n], which by definition contains half of the total salaries, are both greater the "median", but there is another interval [m1,m2] above the median, making more than half the salaries greater than the median, which contradicts the definition of median.

Case 2: The median is contained in the elminiated interval [m2,n]
The intervals [1,m1] and [1,m2], which by definition contains half of the total salaries, are both less the "median", but there is another interval [m1,m2] below the median, making more than half the salaries less than the median, which contradicts the definition of median.

Therefore, the algorithm does not eliminate the median for n=k+1 if it does not eliminate the median for n=k. Therefore the algorithm never removes the median while continuing to reduce n to n/2.

6.

```
int index1=0
int counter1=0
int index2=0
for(int i=0;i<n;i++){
        if(counter1==0){
                index2=index1;
                index1=i
                counter1++;
        }
        else{
                if(equivalent(index,i))
                        counter1++;
                else
                        counter1--;
        }
}
counter1=0;
int counter2=0
for(int i=0;i<n;i++){
        if(equivalent(index,i))
                counter1++;
        if(equivalent(index2,i))
                counter2++;
}
if(counter1>=(n/2) || counter2>=(n/2))
        return true;
else
        return false;
```