Stamatios Morellas
COM S 311 – Homework 3
9/30/19

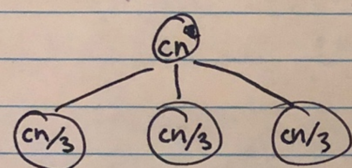
Stamati Morellas

Section 6 - Tuesday 1:10 - John Wahlig

9/30/19

$\boxed{\text{There are } \log_2 n \text{ levels of recursion}}$

① a) $T(n) \le 3T(\frac{n}{2}) + cn^2$; $\underline{q=3}$, $T(2) \le c$



$$T(n) = 3T(\tfrac{n}{2}) + cn^2$$
$$= 3 \cdot \left[3 \cdot T(\tfrac{n/2}{2}) + (\tfrac{n}{2})^2 c\right] + cn^2$$
$$= 3^2 T(\tfrac{n}{2^2}) + \frac{3cn^2}{4} + cn^2$$

$$\cancel{3^2 T(\tfrac{n}{2^2}) + \ldots cn^2}$$

$$T_2(n) = 3^2 T(\tfrac{n}{2^2}) + \frac{7cn^2}{4} \quad \text{(lvl 2)}$$
$$= 3^2 \cdot \left[3 \cdot T(\tfrac{n/2^2}{2}) + cn^2\right] + \frac{7cn^2}{4}$$

$$\cancel{3^2\left[3T(\tfrac{n}{2^3}) + c n^2\right] + \tfrac{7cn^2}{4}}$$

$$= 3^2\left[3T(\tfrac{n}{2^3}) + c(\tfrac{n}{2^2})^2\right] + \frac{7cn^2}{4}$$
$$= 3^3 T(\tfrac{n}{2^3}) + \left(3^2 c \cdot \frac{n^2}{2^4}\right) + \frac{7cn^2}{4}$$
$$= 3^3 T(\tfrac{n}{2^3}) + \frac{9cn^2}{16} + \frac{7cn^2}{4}$$
$$T_3(n) = 3^3 T(\tfrac{n}{2^3}) + \frac{37cn^2}{16} \quad \text{(lvl 3)}$$

$$\cancel{T_k(n) = 3^k T(\tfrac{n}{2^k}) + 3^{k-1}\tfrac{cn^2}{2^{2k-2}}}$$

Recurrence at level k →
$$\boxed{T_k(n) = 3^k T(\tfrac{n}{2^k}) + \sum_{j=1}^{k} \frac{3^{k-1} cn^2}{2^{2k-2}}}$$

$(k = \log_2 n)$
$$\boxed{T(n) \le 3^{\log_2 n} T(\tfrac{n}{2^{\log_2 n}}) + \sum_{j=1}^{\log_2 n} \frac{3^{\log_2 n - 1} cn^2}{2^{2\log_2 n - 4}}}$$

$$T(n) \le 3^{\log_2 n} T(1) + \frac{3^{\log(\log n) - 1} cn^2}{2^{2(\log(\log n)) - 4}}$$

$$\Rightarrow T(n) \le 3^{\log_2 n} T(1) +$$

$$\boxed{\text{Worst-Case Runtime: } \in O(n^2 \log n)}$$

b) $T(n) \le 2T(\frac{n}{2}) + cn\log n$; $T(2) \le c$

$$T_1(n) = 2T(\tfrac{n}{2}) + cn\log n$$
$$= 2\cdot\left[2\cdot T(\tfrac{n/2}{2}) + c\tfrac{n}{2}\log\tfrac{n}{2}\right] + cn\log n$$
$$= 2^2 T(\tfrac{n}{2^2}) + 2\left(c\tfrac{n}{2}\log\tfrac{n}{2}\right) + cn\log n$$
$$= 2^2 T(\tfrac{n}{2^2}) + cn\log\tfrac{n}{2} + cn\log n$$

$$T_2(n) = 2^2 T(\tfrac{n}{2^2}) + 2cn\left(\log\tfrac{n^2}{2}\right)$$
$$= 2^2\cdot\left[2\cdot T(\tfrac{n/2^2}{2}) + c(\tfrac{n}{2^2})\log\tfrac{n}{2^2}\right] + 2cn\left(\log\tfrac{n^2}{2}\right)$$
$$= 2^3\cdot T(\tfrac{n}{2^3}) + cn\log\tfrac{n}{2^2} + 2cn\log\tfrac{n^2}{2}$$

$$T_3(n) = 2^3 T(\tfrac{n}{2^3}) + 3cn\log\tfrac{n^3}{8}$$
$$= 2^3\cdot\left[2T(\tfrac{n}{2^4}) + c\tfrac{n}{2^3}\log\tfrac{n}{2^3}\right] + 3cn\log\tfrac{n^3}{8}$$
$$= 2^4 T(\tfrac{n}{2^4}) + cn\log\tfrac{n}{2^3} + 3cn\log\tfrac{n^3}{2^3}$$

$$T_4(n) = 2^4 T(\tfrac{n}{2^4}) + 4cn\log\tfrac{n^4}{2^6}$$

$$\boxed{T_k(n) = 2^k T(\tfrac{n}{2^k}) + knc\log\tfrac{n^k}{a}}$$

Let $a = \sum_{j=0}^{k} 2^k$

(2)

```
functionDandC(Arr[], int k) {
    int i = 1, j = 0, c = 0;
    while (i < Arr.size - k) {          [O(n - k)]
        c = Arr[i]; //current element
        j = i + k;
        while (j >= 0 and Arr[j] > c) {   [O(n)]
            Arr[j] = Arr[i]; //swap elements
            j--;
        }
        Arr[j+1] = c; i++;
    }
```

~~i = Arr.size - k~~

~~while (i < Arr.size) {~~

~~c = Arr[i]~~

~~j--~~

Two while loops:

Outer runtime: $O(n-k)$

Inner runtime: $O(n)$

$\boxed{\text{Total Runtime: } O(n^2 - nk) \approx O(n^2)}$

```
③ function getPurple(S[]) {
      int x, y, p, q;
      int i, j = i+1; c=0;
      N[] = new empty sets of size S.size;
      while (i < S.size) {    O(n)
            while (j < S.size) {  O(n)

                  x= S[i].getX();
                  y= S[i].getY();
                  p= S[j].getX();
                  q= S[j].getY();
                  if(x < p && y<q){
                        N[c] = S[i];

                        c++;   break;
                  } else { N[c]= S[j]; }
                  j++;
            }
            i++;
      }
      return N[];
}
```

Outer loop runtime: $O(n)$

Inner loop runtime: $O(n)$

Total: $O(n^2)$