Muhammed ÖZKAN 151044084

CSE-321 Introduction to Algorithm Design

Fall 2020 Homework 3

1)

a)

$$T(n) = 27T\left({}^{n}/_{3}\right) + n^2 \qquad a = 27,\ \ b = 3,\ \ d = 2$$

$$b^d = 3^2 = 9 \ \ \rightarrow \ \ 27 > 9: \ \ a > b^d \ \ \rightarrow T(n) \in \Theta(n^{\log_b a})$$

$$T(n) \in \Theta(n^{\log_3 27}) \ \in \Theta(\boldsymbol{n^3})$$

b)

$$T(n) = 9T\left({}^{n}/_{4}\right) + n \qquad a = 9,\ \ b = 4,\ \ d = 1$$

$$b^d = 4^1 = 4 \ \ \rightarrow \ \ 9 > 4: \ \ a > b^d \ \ \rightarrow T(n) \in \Theta(n^{\log_b a})$$

$$T(n) \in \Theta(n^{\log_4 9}) \ \in \Theta(\boldsymbol{n^{\log_2 3}})$$

c)

$$T(n) = 2T\left({}^{n}/_{4}\right) + \sqrt{n} \qquad a = 2,\ \ b = 4,\ \ d = 1/2$$

$$b^d = 4^{1/2} = 2 \ \ \rightarrow \ \ 2 = 2: \ \ a = b^d \ \ \rightarrow T(n) \in \Theta(n^d \log n)$$

$$T(n) \in \Theta(n^{1/2} \log n) \ \in \Theta(\boldsymbol{\sqrt{n} \log n})$$

d)

$$T(n) = 2T\left(\sqrt{n}\right) + 1 \qquad m = logn,\ \ n = 2^m \ \rightarrow \ T(2^m) = 2T\left(2^{m/2}\right) + 1$$

$$let's\ say \qquad T(2^m) = S(m) \rightarrow \ T\left(2^{m/2}\right) = S({}^{m}/_{2})$$

$$S(m) = 2S\left({}^{m}/_{2}\right) + 1 \qquad a = 2,\ \ b = 2,\ \ d = 0$$

$$b^d = 2^0 = 1 \ \ \rightarrow \ \ 2 > 1: \ \ a > b^d \ \ \rightarrow S(m) \in \Theta(m^{\log_b a})$$

$$S(m) \in \Theta(m^{\log_2 2}) \ \in \Theta(\mathrm{m})$$

$$T(2^m) = S(m) \rightarrow \ T(2^m) \in \Theta(\mathrm{m})$$

$$m = logn \rightarrow \ T(n) \in \Theta(\mathbf{logn})$$

e)

$$T(n) = 2T(n-2), \qquad T(0) = 1, \qquad T(1) = 1$$

$$T(n) = 0T(n-1) + 2T(n-2)$$

$$\propto^2 = 0 \propto +2 \quad : characteristic\ equation$$

$$\propto^2 - 2 = 0 = \left(\propto -\sqrt{2}\right)\left(\propto +\sqrt{2}\right), \qquad \propto_1 = \sqrt{2}\ and\ \propto_2 = -\sqrt{2}$$

$$two\ distinct\ and\ real\ root$$

$$T(n) = c_1(\propto_1)^n + c_2(\propto_2)^n$$

$$T(n) = c_1\left(\sqrt{2}\right)^n + c_2\left(-\sqrt{2}\right)^n$$

$$T(0) = c_1 + c_2 = 1$$

$$T(1) = \sqrt{2}\,c_1 - \sqrt{2}\,c_2 = 1$$

$$\sqrt{2} * T(0) = \sqrt{2}\,c_1 + \sqrt{2}\,c_2 = \sqrt{2}$$

$$T(1) = \sqrt{2}\,c_1 - \sqrt{2}\,c_2 = 1$$

$$T(1) + \sqrt{2}\,T(0) = 2\sqrt{2}\,c_1 = \sqrt{2} + 1$$

$$\boldsymbol{c_1 = \frac{\sqrt{2}+1}{2\sqrt{2}}} \qquad \boldsymbol{c_2 = 1 - c_1 = \frac{\sqrt{2}-1}{2\sqrt{2}}}$$

$$\mathrm{T(n)} = \frac{\sqrt{2}+1}{2\sqrt{2}}\,(\sqrt{2})^{\mathrm{n}} + \frac{\sqrt{2}-1}{2\sqrt{2}}\,(-\sqrt{2})^{\mathrm{n}} \rightarrow \mathrm{T(n)} \in \Theta(\boldsymbol{\sqrt{2}^{\mathbf{n}}})$$

f)

$$T(n) = 4T\left({}^{n}/_{2}\right) + n \quad , T(1) = 1 \qquad a = 4, \ \ b = 2, \ \ d = 1$$

$$b^d = 2^1 = 2 \ \rightarrow \ 4 > 2 : \ a > b^d \ \rightarrow T(n) \in \Theta(n^{\log_b a})$$

$$T(n) \in \Theta(n^{\log_2 4}) \ \in \Theta(\boldsymbol{n^2})$$

g)

$$T(n) = 2T\left(\sqrt[3]{n}\right) + 1\,, T(3) = 1 \qquad m = logn,\ \ n = 2^m \ \rightarrow\ \ T(2^m) = 2T\left(2^{m/3}\right) + 1$$

$$let's\ say \qquad T(2^m) = S(m) \rightarrow\ T\left(2^{m/3}\right) = S(m/3)$$

$$S(m) = 2S\left(m/3\right) + 1 \qquad a = 2,\ \ b = 3,\ \ d = 0$$

$$b^d = 3^0 = 1 \ \rightarrow\ \ 2 > 1:\ \ a > b^d \ \ \rightarrow S(m) \in \Theta(m^{\log_b a})$$

$$S(m) \in \Theta\left(m^{\log_2 3}\right) \in \Theta(\mathrm{m})\ , \qquad \log_2 3 < 1$$

$$T(2^m) = S(m) \rightarrow\ T(2^m) \in \Theta(\mathrm{m})$$

$$m = logn \rightarrow\ T(n) \in \Theta(\mathbf{logn})$$

2)

$$T(n) = n * T\left(n/2\right), \qquad T(1) = 1$$

$$= n * \frac{n}{2} *\ T\left(n/4\right)$$

$$= n * \frac{n}{2} * \frac{n}{4} *\ T\left(n/8\right)$$

$$= n * \frac{n}{2} * \frac{n}{4} * \frac{n}{8} *\ T\left(n/16\right)$$

$$= n * \frac{n}{2} * \frac{n}{4} * \frac{n}{8} * \frac{n}{16} *\ T\left(n/32\right)$$

$$= \frac{n^k}{2^{\frac{k(k-1)}{2}}} * T\left(\frac{n}{2^k}\right) \qquad , \quad n = 2^k \rightarrow logn = k$$

$$= \frac{(2^k)^k}{2^{\frac{k^2-k}{2}}} * T(1) = \ \frac{2^{k^2}}{2^{\frac{k^2-k}{2}}} \ = \ 2^{k^2} * 2^{\frac{k-k^2}{2}} = 2^{\frac{2k^2+k-k^2}{2}} = 2^{\frac{k^2+k}{2}}$$

$$= 2^{\frac{log^2n + logn}{2}} = 2^{\left(\frac{log^2n}{2} + \frac{logn}{2}\right)}$$

$$= 2^{\frac{log^2n}{2}} * 2^{\frac{logn}{2}} = 2^{\frac{(\log_2 n)^2}{2}} * \sqrt{n}$$

$$\boxed{2^{\frac{1}{2}*\log_2 n * \log_2 n} = \left(2^{\log_2 n}\right)^{\frac{1}{2}\log_2 n} = n^{\frac{1}{2}\log_2 n}}$$

$$= n^{\frac{1}{2}\log_2 n} * n^{\frac{1}{2}} = n^{\frac{1}{2}(\log_2 n+1)} = n^{\frac{\log_2 n+1}{2}}$$

$$T(n) \in \mathbf{O}(\boldsymbol{n}^{\frac{\log_2 n+1}{2}})$$

The program prints as many lines as the value opposite the value of n in the table below.

2	**2**
4	**8**
8	**64**
16	**1024**
32	**32768**
64	**2097152**
128	**268435456**
256	**68719476736**
.....	**.....**

3)

$$T(n) = 3T\left({}^{2n}/_{3}\right) + 1, \qquad T(1) = 1$$

$$T(n) = 3T\left({}^{2n}/_{3}\right) + 1$$

$$= 3\left[3T\left({}^{2^2 n}/_{3^2}\right) + 1\right] + 1 \quad = \quad 3^2 T\left({}^{2^2 n}/_{3^2}\right) + 3 + 1$$

$$= 3^3 T\left({}^{2^3 n}/_{3^3}\right) + 3^2 + 3 + 1$$

$$= 3^k T\left({}^{2^k n}/_{3^k}\right) + 3^{k-1} + \cdots + 3^2 + 3 + 1$$

$$Let\ n = \frac{3^k}{2^k} \ , \ n = \left(\frac{3}{2}\right)^k$$

$$= 3^k T({}^{n}/_{n}) \underbrace{+ 3^{k-1} + \cdots + 3^2 + 3 + 1}$$

$$= 3^k + 3^k = 2 * 3^k$$

$$T(n) = 2 * 3^k \approx \log_3 n$$

$$T(n) \in \mathbf{O}(\mathbf{log_3}\,\boldsymbol{n})$$

4)

Implementation of Insertion Sort and Quick Sort:

```
def insertion_sort(lst):
	for i in range(1, len(lst)):
		item = lst[i]
		sorted_idx = i − 1
		while sorted_idx >= 0 and lst[sorted_idx] > item:
			lst[sorted_idx + 1] = lst[sorted_idx] # swap operation performed
				sorted_idx −= 1
	lst[sorted_idx + 1] = item
```

```
def quick_sort(lst, low_idx, high_idx):
	if low_idx < high_idx:
		partition_idx = partition(lst, low_idx, high_idx)
		quick_sort(lst, low_idx, partition_idx − 1)
		quick_sort(lst, partition_idx + 1, high_idx)

def partition(lst, low_idx, high_idx):
	smaller_idx = low_idx − 1
	pivot = lst[high_idx]
	for j in range(low_idx, high_idx):
		if lst[j] < pivot:
			smaller_idx = smaller_idx + 1
			swap(lst, smaller_idx, j)
	swap(lst, smaller_idx + 1, high_idx)
	return smaller_idx + 1
```

Analyze the average-case complexity:

Let's analyze insertion sort:

$$E\left[I_{i,j}\right] = \frac{1}{2}$$

$$E[I] = \sum_{i<j} E\left[I_{i,j}\right]$$

$$= \sum_{i<j} \frac{1}{2}$$

$$= \frac{1}{2}\binom{n}{2}$$

$$\text{Hence, } \mathbf{T}(\mathbf{n}) \in \mathbf{O}(\mathbf{n}^{\mathbf{2}})$$

Let's analyze quick sort. After the do the partitioning, we recursively call our algorithm on two partitioned lists. Regarding to Hoare Partition algorithm, Since i is between 0 to n − 1, average value of T(i) is:

$$E(T(i)) = \frac{1}{n}\sum_{j=0}^{n-1} T(j)$$

$$E\big(T(n-i)\big) = E\big(T(i)\big) \therefore T(n) = \frac{2}{n}\left(\sum_{j=0}^{n-1} T(j)\right) + cn$$

$$nT(n) = 2\left(\sum_{j=0}^{n-1} T(j)\right) + n^2 \qquad\qquad \mathit{Multiply\ by\ n}$$

$$(n-1)T(n-1) = 2\left(\sum_{j=0}^{n-1} T(j)\right) + (n-1)^2 \qquad\qquad \mathit{put\ n-1\ for\ n}$$

$$nT(n) = (n+1)T(n-1) + 2cn$$

Solving this recurrence relation:

$$\frac{T(n)}{n+1} = \frac{T(n-1)}{n} + \frac{2c}{n+1}$$

$$\frac{T(n-1)}{n} = \frac{T(n-2)}{n-1} + \frac{2c}{n}$$

$$\frac{T(n)}{n+1} = \frac{T(1)}{2} + 2c\sum_{j=3}^{n+1}\frac{1}{j}$$

$$\sum_{j=3}^{n+1}\frac{1}{j} \to \ln n + \gamma \therefore \frac{T(n)}{n+1} = \frac{T(1)}{2} + 2c\ln n + 2c\gamma$$

$$\text{Hence, } \mathbf{T(n)} \in \mathbf{O(nlog\,\boldsymbol{n})}$$

Comparative analysis of both sorting algorithms:

Theoretically quick sort is better, since $T_{qui}(n) \in O(nlogn)$ and $T_{ins}(n)O(n^2)$. And with experiments, I have added a counter in python code to a bunch of relevant places to see actually which one is better in practically:

```
def swap(lst, idx1, idx2, count=0):
        count += 1
        temp = lst[idx1]
        lst[idx1] = lst[idx2]
        lst[idx2] = temp
```

If we are to count the number of swap operations in the sorting algorithms respectively, insertion sort performs more swap operations:

Same arrays sorted via Insertion Sort, Count : 20, 36, ...

Same arrays sorted via Quick Sort, Count : 6, 13, ...

Similar patterns observed during the experiments. Therefore we can say, quick sort is both better in theory and practice.

5) My choice between the three algorithms will be c. Because it combines solutions in linear time. Its difference from others is that the merging process is faster than others.

a)

$$T(n) = 5T\left({}^{n}\!/_{3}\right) + O(n^2) \qquad a = 5,\ \ b = 3,\ \ d = 2$$

$$b^d = 3^2 = 9 \ \ \rightarrow \ \ 5 < 9 : \ \ a < b^d \ \ \rightarrow T(n) \in \Theta(n^d)$$

$$T(n) \in \Theta(\boldsymbol{n^2})$$

b)

$$T(n) = 2T\left({}^{n}\!/_{2}\right) + O(n^2) \qquad a = 2,\ \ b = 2,\ \ d = 2$$

$$b^d = 2^2 = 4 \ \ \rightarrow \ \ 2 < 4 : \ \ a < b^d \ \ \rightarrow T(n) \in \Theta(n^d)$$

$$T(n) \in \Theta(\boldsymbol{n^2})$$

c)

$$T(n) = T(n-1) + n$$

$$T(2) = T(1) + 2$$

$$T(3) = T(2) + 3 = T(1) + 2 + 3$$

$$T(4) = T(3) + 4 = T(1) + 2 + 3 + 4$$

$$T(5) = T(4) + 5 = T(1) + 2 + 3 + 4 + 5$$

$$T(n) = T(n-1) + n = T(1) + 2 + 3 + 4 + 5 + \cdots + n$$

$$if\ we\ assume\ T(1) = 0$$

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

$$T(n) \in \Theta(\boldsymbol{n^2})$$