DAA

__/__/____

K + L - M * N + (O ^ P) * W / U / V * T + Q

Input sb.	Stack	Postfix exp.
K		K
+	+	K
L	+	KL
-	-	KL+
M	-	KL+M
*	-*	KL+M
N	-*	KL+MN
+	+	KL+MN*-
(, O	+(	KL+MN*-O
^, P	+(^	KL+MN*-OP
)		KL+MN*-OP^+
*	+*	KL+MN*-OP^
W	+*	KL+MN*-OP^W
/	+/	KL+MN*-OP^W*
U	+/	KL+MN*-OP^W*U
/	+/	KL+MN*-OP^W*U/
V	+/	KL+MN*-OP^W*U/V
*	+*	KL+MN*-OP^W*U/V/
T	+*	KL+MN*-OP^W*U/V/T
+	+	KL+MN*-OP^W*U/V/T*+
Q	+	KL+MN*-OP^W*U/V/T*+Q+

__/__/____

$$T(n) = (C_1 . n) + (C_2 . (n-1)) + (C_3 . (n-1)) + (C_4 (\Sigma t_i)) + (C_5 (\Sigma t_i - 1)) + (C_6 (\Sigma t_i - 1)) + (C_7 . (n-1))$$

Best Case $\quad C_4 * \sum_{i=2}^{n} (t_i) \rightarrow \sum_{i=2}^{n} 1$

$(n-1)$

$$T_n = (C_1 . (n)) + (C_2 . (n-1)) + (C_3 . (n-1)) + (C_4 . (n-1)) + (C_7 . (n-1))$$

$$n(C_1 + C_2 + C_3 + C_4 + C_7) - (C_2 + C_3 + C_4 + C_7)$$

Worst Case $\quad \sum_{i=0}^{n} ti \quad 1 + 2 + 3 : \ldots (n-1)$

$$\sum_{i=0}^{n} =$$

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

For $[C_5 . (T_i - 1)]$ = It will run 1 time for first numbr

$1 - 1 = 0$, that means 0

$0 - - - (n-1)$

We can apply $\frac{n(n-1)}{2}$

$$T(n) = C_1 . n + C_2 . (n-1) + C_3 (n-1) + C_4 \left[\frac{n(n+1)}{2} - 1\right] +$$

$$C_5 \left[\frac{n(n-1)}{2}\right] + C_6 \left[\frac{n(n-1)}{2}\right] + C_7 (n-1)$$

USA

```
p=0
for(i=1 ; p<=n ; i++){
    p=p+i }
```

$O\sqrt{n}$

i	p
1	0+1
2	0+1+2
3	0+1+2+3
4	0+1+2+3+4
5	0+1+2+3+4+5

$\frac{k(k+1)}{2}$

Dense - More connections are there

Sparse - Less connection are there

Complexity classes $g(n)$? $f(n)$?

Insertion Sort

	Algorithm	Cost	Time
①	Repeat through step 7 for i=2,3,...n	C_1	n
②	Set t ← A[i]	C_2	$(n-1)$
③	set p ← p←i - 1	C_3	$(n-1)$
④	Repeat while (p > 0 and A[p]) through step 6	C_4	$\sum_{i=2}^{n} t_i$
⑤	Set A[p+1] ← A[p]	C_5	$\sum_{i=2}^{n}(t_i-1)$
⑥	Set p ← p-1	C_6	$\sum_{i=2}^{n}(t_i-1)$
⑦	Set A[p+1] ← t	C_7	$(n-1)$

AAD

2 3 1 * + 9 -

Symbol Scanned	Stack	
2	2	
3	23	
1	231	
*	2 [3*1]	=> 23
+	2+3	=> 5
9	59	
-	5-9	=> 4

```
for (i=0; i<N; i++)   ← it will run (n+1) times including false statement
  {
  statement;          ← (n) times
  }
```

Order of n | O(n) → $f(n) = 2n + 1$

---

if $\left(\frac{n}{2} + 1\right)$ then → O(n)

```
for (i=0; i<n; i++){
    for (j=0; j<i; j++){
        stmt;
    }
```

$0 + 1 + 2 + 3 + 4 + 5 \cdots n$

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

$O(n^2)$

__/__/____

if $\log f(n) = O(\log(g(n))) \Rightarrow f(n) = O\,g(n)$
if $\log f(n) = \omega(\log(g(n))) \Rightarrow f(n) = \omega\,g(n)$

Q> Constant are removed while finding asymptotic

$f(n) = 2^n$
$\log 2^n$
$n \log 2$

$g(n) = n^5$
$\log n^5$
$5 \log n$

~~$f(n) < g(n)$~~
$n > \log n$
$f(n) > g(n)$
$2^n = \omega\, n^5$
$f(n) < g(n)$

Log approach is applicable where power is there

Q>

2^n | 2^{2n}
$\log 2^n$ | $\log 2^{2n}$
$n \log 2$ | ~~2~~ $2n \log 2$ ← Constants are removed
n | $2n$

$n \simeq n$

$f(n) < g(n)$
$f(n) = O\,g(n)$
$f(n) > g(n)$
$f(n) = \omega\, g(n)$

Asymptotic mean there is no space for constant

$\lim_{n \to \infty} \frac{1}{2^n} = \frac{1}{2^\infty} = \frac{1}{\infty} = 0$

DO Not use log with Big O or Θ (theta)

$g(n) > f(n)$ / $g(n) < f(n)$ → That means rapid growth is there
If (equal to) is used then it is not

$$\frac{pn - p + 2n - 2np}{2} = \frac{-np - p + 2n}{2}$$

Q) Determine for a fixed n for the value of P for which we get max value of cost of avg & min val of cost avg.

Unsuccessful search = P (Probability) = 0

$$\frac{-n(0) - 0 + 2n}{2} = \frac{2n}{2}$$

$P = 1, \frac{n}{2}$

Q) How many comparisons, both successful and unsuccessful with which we made with brute algorith in searching for each of the following method in binary text. 6000 zeros and patterns are ① 00001
② 10000
③ 01010

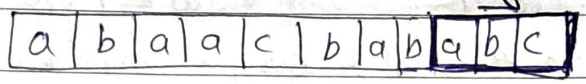
Naive algorithm => $(n - m + 1)$

① 596 * 5
② 596 * 1 (because loop will run only once, inner loop)
③ 596 * 2

UNIT - 2 (Searching Algorithm)

This unit consist of Hybrid searching Algo.

Sentential form



we extend the array insted of making new array we need single space now.

String matching problem

Normal sequential approach

a b c a c d e
a b c
a c a
c a c
c d e
d e
e

[Naive - string - Matching]

```
n ← length [T]
m ← length [P]
for s ← 0 to n-m
    do if P[1...m] = T[s+1...s+m]
    then print ("Pattern occurs")
```

$O(n-1)$

If P ← Probability of successful search

$1-P$ ← Probability of unsuccessful search

Average case

$$\left(1 \times \frac{P}{m}\right)\left(2 \times \frac{P}{m}\right) \cdots \left(\frac{(n-1) \times P}{n}\right) + n(1-P)$$

[Successful] + [Unsuccessful]

$$\frac{P}{n} \cdot \frac{n(n-1)}{2} + n(1-P) \Rightarrow \frac{P(n-1) + 2n + 2np}{2}$$

Consider the following function :-

$f(n) = 2^n$ $\quad g(n) = n!$ $\quad h(n) = n \log n$

(a) $f(n) = O\, g(n)$ T $\quad g(n) = O\, h(n)$ F

(b) $f(n) = \Omega\, g(n)$ F $\quad g(n) = O\, h(n)$ F $\quad n \log n < 2^n < n!$

(c) $g(n) = O\, f(n)$ F $\quad h(n) = O\, f(n)$ T

(d) $h(n) = O\, f(n)$ T $\quad g(n) = \Omega\, f(n)$ T

Arrange all time function in ascending & ~~dd~~ decending

$f(n) < g(n)$ $\quad f(n) \leq g(n)$ $\quad g(n) \leq h(n)$

$f(n) > g(n)$ $\quad 2^n \leq n!$ $\quad n! \leq n \log n$

$f(n) \geq g(n)$ $\quad g(n) \leq h(n)$

$2^n \geq n!$ $\quad n! \leq n \log n$

---

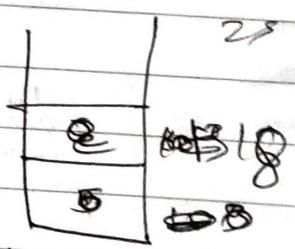
what is the

5 2 * 3 3 2 + * +

? Full & Complete ~~BT~~

? arity operation

? Parse expression (Scanning)



Pre-Post use to parse without parantheses.

In ~~with~~ parse with parantheses.

In division ~~tree~~ making tree is used.

__/__/____

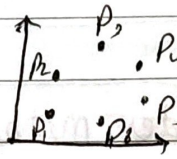
Brute force

A straight forward approach, based on direct approach

↓

No Restriction

Closest Pair Problem



$d_{min} \leftarrow \infty$ $O(n^2)$

for $i \leftarrow 1$ to $n-1$ do

  for $j \leftarrow i+1$ to n do

    $d \leftarrow sqrt((x_i - x_j)^2 + (y_i - y_j)^2)$

Convex Hull Problem

Exhaustive Search

- Potential solution
- Feasible / or not
- Solution

Traveling Salesman Problem.

$\Phi((n-1)!)$

In brute force we will traverse every solution even after we find solution.

then "Pattern occur with shift" s

if $s < n - m$

then $t_{s+1} \leftarrow (d(t_s - T[s+1]h) + T[s+m+1] \bmod q$

Hashcode ↓ Hash function

Q) 13240128

401

13[2] = 401

[32]4 = 401

$\frac{1*10^2 + 3*10^1 + 2*10^0}{8*10^2 + 2*10 + 4*10^0} \Rightarrow 10[3 \times 10^1 + 2*10] + 4*10^0$

$10[t_0 - 1*10^2] + 1*10$

$t_{s+1} = d[t_s - T(s+1)*h$

__/__/____

Rabin-Karp Algorithm

Q) When this algo. is applicable?
This algo is applicable where some pattern is repeated again and again.
Rabin-Karp uses mod value with prime number.

x = Length of pattern

x mod (prime)

Suprious hit - When we get same hash value but ~~value~~ is different (text not match)

Q) Why we take prime number?
If we take normal random number then more chances of suprious hits are there, but not with prime number.

- Prime number giving the largest value will be selected

Hash code we use is known a Rolling Hash Function

```
n ← length[T]
m ← length[P]
h ← d^(m-1) mod q   // d = Radix , q = Prime
p ← 0 ; t0 ← 0
for i ← 1 to m
    do p ← (dp + P[i]) mod q
       t0 ← (dt0 + T[i]) mod q
for s ← 0 to n - m
    do if p = ts
          then if P[1...m] = T[s+1...s+m]
```

↳ Base value, what type value are passing
decimal 10, hexadecimal = 16

Case 3: If $\log_b a < k$,

If $p >= 0$, $T(n) = \Theta(n^k \log^p n)$

If $p < 0$, then $T(n) = \Theta(n^k)$

Eg > $T(n) = 2T\left(\frac{n}{2}\right) + n^2 \log^{-1} n$

$\frac{n^2}{\log n}$

$a = 2$
$b = 2$
$k = 2$
$p = -1$

Case 3, 2nd part will be used

Masters Method for Decreasing

Condition: ~~a>0~~, $b > 0$, $k \geq 0$

Case 1: If $a < 1$, $T(n) = \Theta(n^k)$
Case 2: If $a == 1$, $T(n) = \Theta(n^{k-1})$
Case 3: If $a > 1$, $T(n) = \Theta(n^{n/b} * f(n))$

Merge Sort with Master Theorem

$T(n) = 2T\left(\frac{n}{2}\right) + n$

$a = 2$ $b = 2$
$c = 1$ $d = 0$

↓

$n^k \log^{p+1} n$

↓

$n \log n$

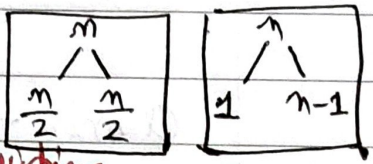
__/__/____

Knapsack Problem by Exhaustive search.

The assignment Problem

Master Method for Dividing Function

$$T(n) = aT(n/b) + f(n)$$



Dividing Master method is used.

Decreasing

where $f(n) = \Theta(n^k \log^p n)$

Eg ▶ $T(n) = 2T(n/2) + n^2$ (k)

$a = 2$
$b = 2$
$k = 2$

To apply master theorem there are 3 condition to satisfy :

- $k \geq 0$
- $a \geq 1$
- $b > 1$

There are 3 cases :—

We will compare 2 value by ~~log~~ $\log_b a$

Case 1: If $\log_b a > k$

- $T(n) = \Theta(n^{\log_b a})$

Case 2: If $\log_b a == k$

- → If $p > -1$, $T(n) = \Theta(n^k \log^{(p-1)} n)$
- → If $p = -1$, $T(n) = \Theta(n^k \log \log n)$
- → $p < -1$, $\Theta(n^k)$

__/__/__

Counting Sort

2	5	3	0	2'	3'	0'	3''

① We will make new array $C[0...k]$

② for $i = 0 \rightarrow k$
$C[i] = 0;$

③ for $j = 1$ to n
$C[A[j]] = C[A[j]] + i$

④ for $i = 1 \rightarrow k$
$C[i] = C[i] + C[i-1]$

⑤ for $i = n$

Radix Sort Pick least significat first and arrang them on list.

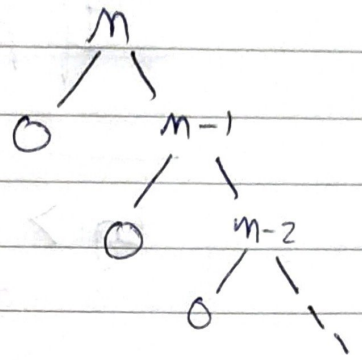
Q) Perform multiplication of given large integer
~~9574~~ 957, 9873

__/__/____

Reccurence eq. for Merg Sort

$T(n) = 2T(\frac{n}{2}) + \Theta(n), \quad T(1) = 0$

Quick Sort worst case =>



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

$a = 1$

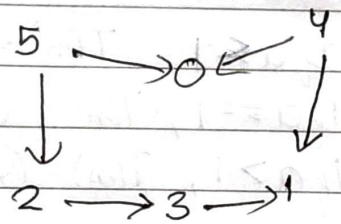
$b = 1$

$k = 1$

Bucket Sort $O(n+k)$ Time complexity

Space complexity $O(\text{max element})$

Topological sort



Step 1

$0 \rightarrow$

$1 \rightarrow$

$2 \rightarrow 3$

$3 \rightarrow 1$

$4 \rightarrow 0, 1$

$5 \rightarrow 2, 0$

Stack → | 0 | 1 | 3 | 2 | 4 | 5 |

↳ we add 3 first because 1 was dependent on 2 (DFS)

- we will fill each element 1 by 1

In graph => Time complexity = $O(v+e)$

vertex + edge