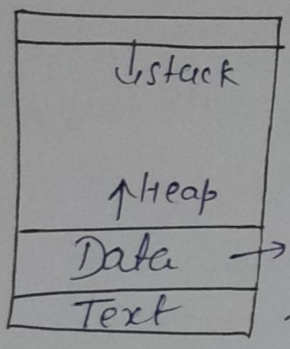
Doubt Class1 Recursion

Date-09/10/2023



OS me koi bhi process bnti hai toh by default usko stack memory de jaati hai 8kB - 8MB

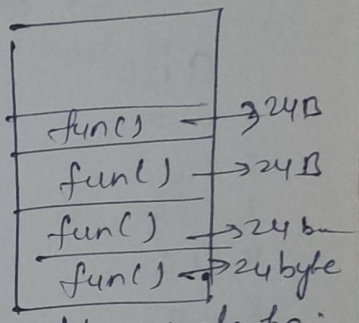
Jaise hi bhar jayegi waise hi stack overflow.

Heap → Dynamic Memory allocation

Toh recursion me base case isliye important hai taaki stack overflow se bach sake

Suppose ek func" likha hai

```
fun(int a)
  if (a>1000) return;
  vector<int> v = {1,2,3,4,5};
  fun(++a);
}
```

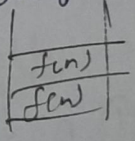


allowed stack size by OS

[24×4] 96B → segmentation fault

Stack bhar jaata hai

Stack overflow.



Time Complexity →

1. O(n)
2. $O(2^n)$ exponential means sol^n bakwas hai → need of optimization
3. O(log n)

RE = Space Complexity = No of stack block taken calls.

or no of calls → jab tak reach par base case pe.

All posibilities ki baat aaye Thik ki recursion lagana hai.

① Binary Search which is best

Iterative ✓ Recursive

Prefer because stack space nhi leta hai O(1)

Recursive → O(log n)

* Subsequence of String →

Pattern include / exclude Pattern

↳ New Approach w/o Recursion

abc

1 ""
2 a
3 ab
4 ac
5 b
6 bc
7 c
8 abc

n=3 2^n ⇒ no of subsequence

	A	B	C				$2^3 = 8$
0 →	0	0	0	–	–	–	""
1 →	0	0	1	–	–	C	C
2 →	0	1	0	–	B	–	B
3 →	0	1	1	–	B	C	BC
4 →	1	0	0	A	–	–	A
5 →	1	0	1	A	–	C	AC
6 →	1	1	0	A	B	–	AB
7 →	1	1	1	A	B	C	ABC

mapping.

* Algorithm.

① n = s.size()

② no. of subsequence = 2^n ⇒ NOS

③ for(i=0; i < 2^n; i++) NOS.

3.1) Convert i to Binary → % 2
→ >>

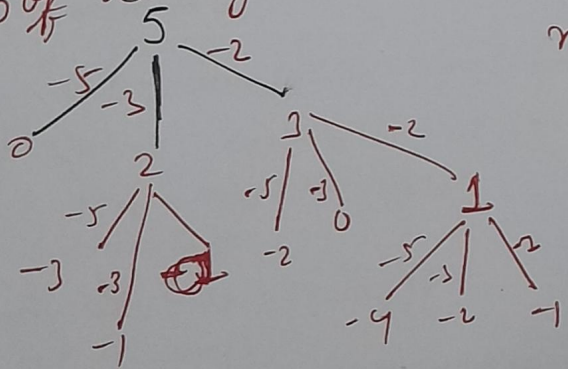
3.2) Jab Jab '1' → char to be taken

```
#include<iostream>
void Subseq(string &s)
    int num_seq = pow(2,s.size());
  for(int n=0; n<num_seq; ++n){
     string out;
     int num=n;
     // convert num to binary
    int i=0;
    while(num){
      int b = (num & 1);
      cout<<b<<" ";
      num>>=1;
    }
    cout<<endl;
}
int main(){
 string="abc";
  Subseq(s);
  return 0;
}
```

```
int i=0;
  while(num){
    int b = (num & 1);
    if(b)
      out.push_back(s[i]);
      ++i;
    num>>=1;
  }
  cout<<out<<endl;
}
```

Iterative approach is better.

Ques → Cut Segments :- Find.
Maxm no of cut segments.




```
    |
  0 |
  | |
  0 0 1
  | 0 1
  0 | 1
  | | |
```

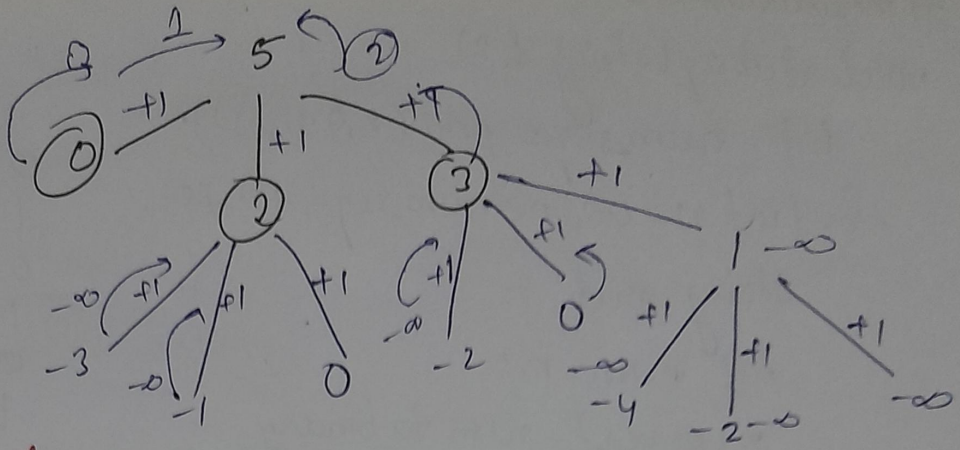
dry run →

a	b	c
0	1	2

num=1
1 i=0 → a
1 right shift → 0.

num=2
first bit=0, & i=0;
i++.
seco. bit=1 & i=2
→ s[i]=b.

num=5
first bit=1 ⇒ i=0 → s[0]=a
next 0 ⇒ i=1 → s[1] X
next 1 ⇒ i=2 → s[2]=c



(4) Coin change:-

amount = 11

coins = 1, 2, 5