Lab Session 2

Sunday, 7 March 2021 10:07 AM

1) → SIXLETS ✓
2) → Palindrome Partitioning
3) → Combinations Sum
4) → {Remove Invalid Paranthesis} → YES
5) → Max Consecutive one's
6) → Max Rectangular area
7) → Quick Select
8) Count Divisors / Square Sum

1) Sixlet's : Number of Subsets of size B having ≤ 1000

Given N array element find no: of subsets of size B having ≤1000

Hint: {Generate all subset sums & check ≤ 1000}

Back Tracking : How many functions : → 2

parameters

[ar[], i, N, sum, cnt, B]

[array, i^{th} index, N, current subsets size]

required subset size — return

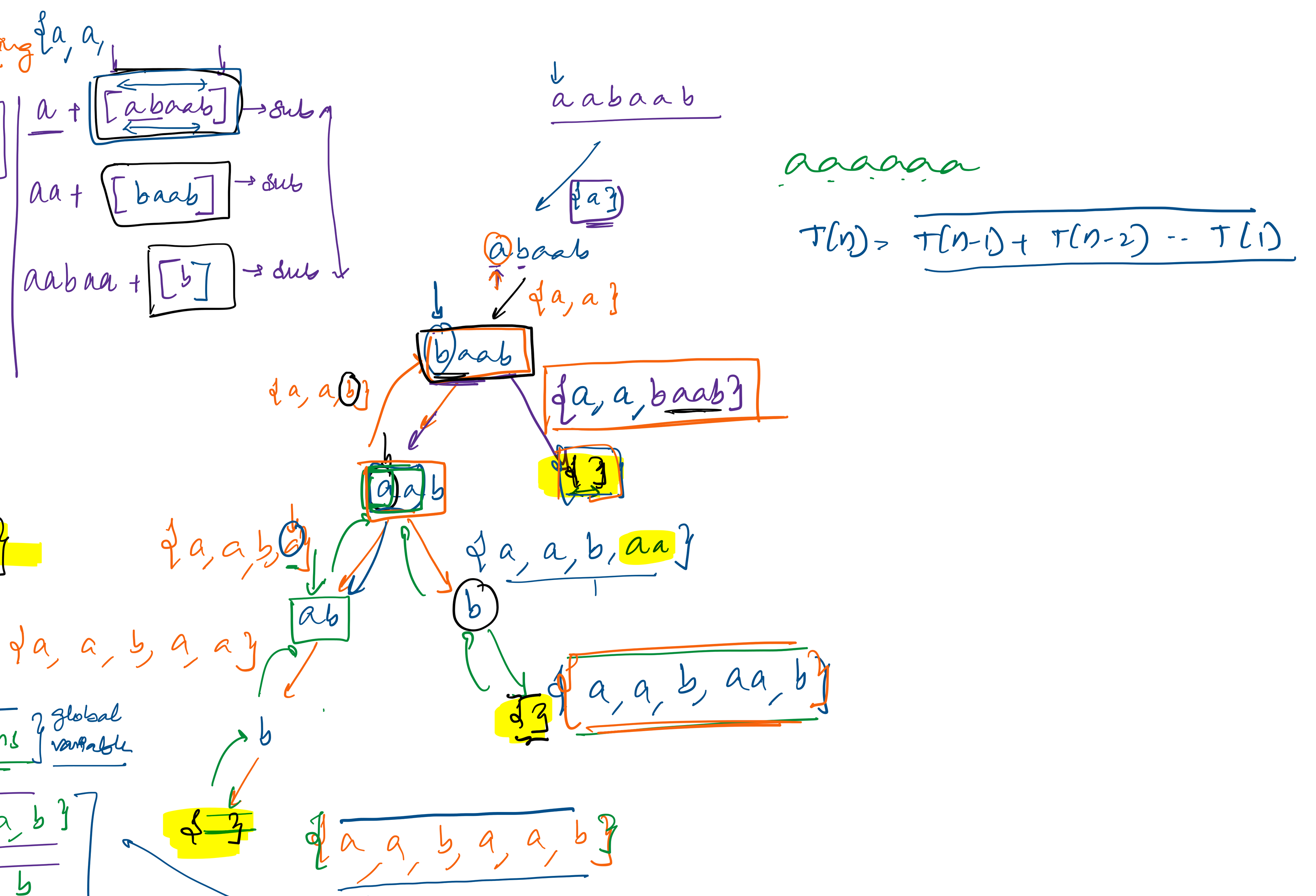
```
if(sum > 1000) return 0;
// if(cnt == B){ return sum <= 1000; }
if( i == N){
    if( sum <= 1000 && cnt == B){
        return 1;
    }
    return 0;
}
{ func(ar, i+1, N, sum, cnt, B) +
  func(ar, i+1, N, sum + ar[i], cnt+1, B) }
```

1001 − 2

(1001) {1−2} {1001 − 2}

Palindrome Partitioning {a, a,

→ a a b | a a b



→ {a, a, baab}

We should also pass list of strings.

→ Parameters

(A) {list of strings}

length of string

aaaaaa

T(n) = T(n−1) + T(n−2) ... T(1)

Final Ans global variable

{a, a, b, a, a, b}
a, a, b aa, b
a, a, baab

Make sure to clear it after before every funcall.

```
vector<vector<string>> ans2;
allper(string A, vector<string> &ans){   // pass by reference
    if(A.size()==0){
        ans2.push_back(ans);
        return;
    }
    for(int i=0; i<A.size(); i++){
        string str = A.substr(0,i);   // you are extracting substring of A from [0-i]
        if(palindrome(str)){
            ans.push_back(str); // 0 i
            // call recursive function call i+1 - n
            string str2 = A.substr(i+1, n-1);
            allper(str2, ans);
            ans.pop_back(); removing last element
        }
    }
}
```

Not exact substring function. Please check it the actual predefine one

Please define your own palindrome function

4Q) Given a paranthesis at max we can remove B, '(' ')' get all balanced paranthesis of max length you can get

Ex: ( )) a (( )

len=5 {( ) a ( )}

Constraints: str ≤ 20 < Total Subsets $2^{20} \approx 10^6$

Importance of Constraints

→ Max iterations : 10^8 / 10^7

BruteForce:

→ Simply generate all subsets & check if Balanced or not

Note: If we are at a i^{th} character which is not '(' ')' in that case you cannot remove that character

Global Variables:

list of strings
vector<string> ans2;

Set<string> ans2 → Note1

Parameters:

A, i, cdiff, ans (temp str)

A: entire string
i: current index position
cdiff: Diff of count of no: of open − no: of closed
ans: All string where we are constructing our balanced paranthesis

⇒ Ex:

(( )

( ) ✓   ( ) ✓

// Note: In required question repetition not allowed. [That means we should store all our answers in a set<strings>?]

[It will take only unique elements]

Ex2:

(( )))

(( ))

( )

→ [Choose only those balanced paranthesis of maxlength]

4 steps:

1) Use back tracking we are generating all balanced paranthesis of all lengths & storing in a set<strings> to avoid duplicates, & also getting len of max balanced paranthesis

2) Iterate on set of a string is having len = (max length) in that can add it in the final ans

3) return final ans

```
[set<string> ans2;
 int len = 0]

void bal(string S, int i, int cdiff, string ans);   // Call by Value
    if( i == A.size()){
        if( cdiff == 0){ // That means we have balanced
            ans2.push_back(ans);; len = max(len, ans.size())
        }
        return   ↳ ans2 is a set use insert( )
    }
    if(A[i] != '(' && A[i] != ')'){ // i-th we have to pick
        ans += A[i];
        bal(A, i+1, cdiff, ans);   ] we are picking data at i-th pos
        return;
    }
    // i-th place decided to skip
    bal(A, i+1, cdiff, ans);
    // i-th place decide to take
    if(A[i] == '('){
        ans += '(';
        bal(A, i+1, cdiff+1, ans)
    }
    else if(cdiff > 0){
        ans += ')';
        bal(A, i+1, cdiff-1, ans)
    }
}
```

((↳ c = 2

```
All solutions :: solve(string A){   → list of strings
    ans2.clear();   ] Initializing Global Variables
    len = 0;
    [string a = "";
     bal(A, 0, 0, a);]  → Before function call
    // we have data in ans2 & also len
    vector<string> ans;
    // Iterate on set string by string, if a string length == len, we
    // can simply add this in our final answer <ans>
    for(auto it = ans2.begin(); it <= ans2.end(); it++){
        if((*it).size() == len){
            ans.push_back(*it);
        }
    }
    return ans;
}
```