Class 83

13 May 2023 11:09

Class 83

Dynamic Programming

ATLASSIAN → OA → LIS → (O(nlogn))

O(n²)

[10,9,2,5,3,7,101,18]

vector<int> dp

dp[i] =

ith element

For(i=0; i<n; i++) → O(n)

(0 - i-1)

↓

increase

For(j=0; j<i; j++)

n

O(n)

O(logn)

n

For loop

10 2 2 4 4

2, 5, 3, 7, 101, 18

dp[0] = 1

lower_bound → O(logn)

O(n)

3

Candidates map

keep value

sorted value

map

2 → 1

3 → 2

7 → 3

18 → 4

3, 4 - 9

→ sorted keys, sorted value

→ same advantage, smaller keys

w → adv

18

y → dp[w] + 1

y > w

2 → 1

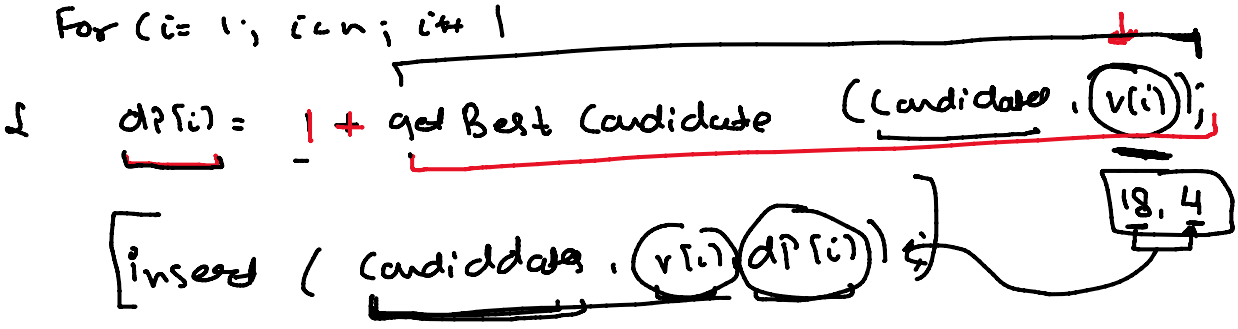
18, 4

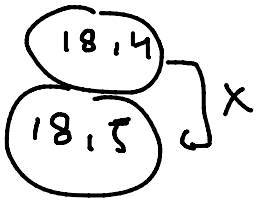
3, 4, 7, 18 . . .

5, 3

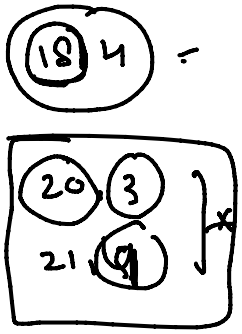
3, 3

```
vector<int> dp(n);
map<int,int> candidates;
dp[0] = 1;
candidates[v[0]] = 1;

For(i=1; i<n; i++)
{   dp[i] = 1 + getBestCandidate(candidates, v[i]);
    insert(candidates, v[i], dp[i]);
}

return *max_element(dp.begin(), dp.end());
}

int getBestCandidate(map<int,int>&candidates, int v)
{   auto it = candidates.lower_bound(v);
    if(it == candidates.begin())
    {  return 0;
    }
    it--;
    return it->second;
}

void insert(map<int,int>&candidates, int v, int adv)
{   if(candidates[v] >= adv)
```



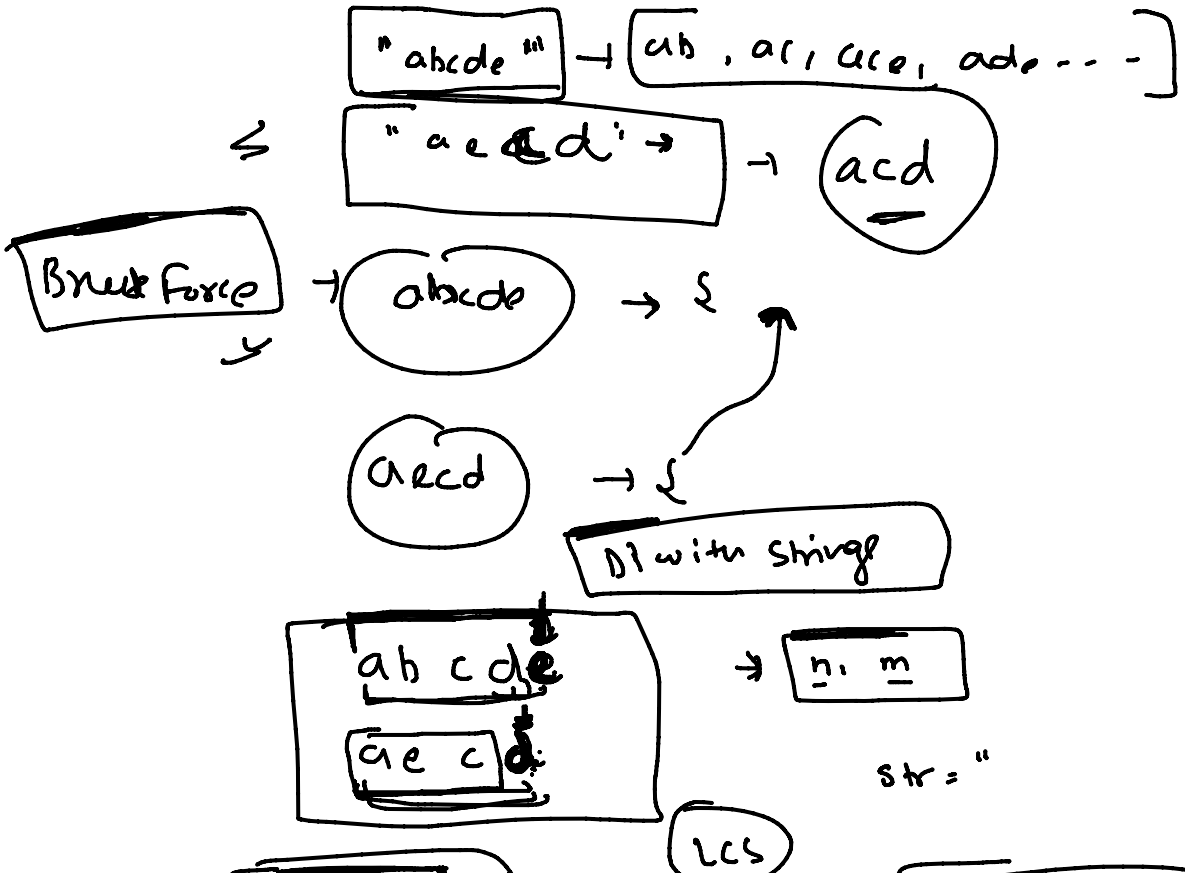


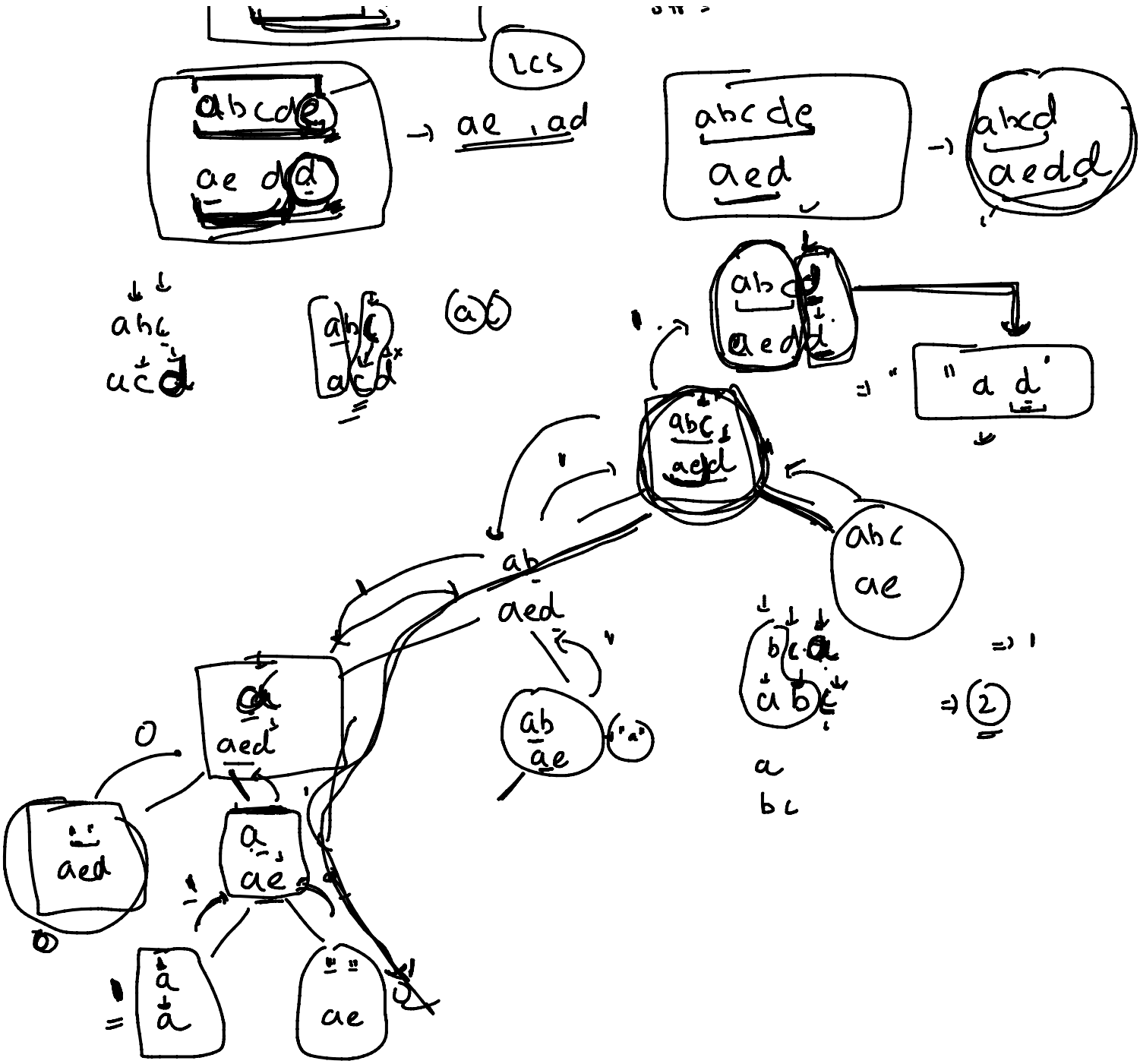
```
if ( candidates[v] ≥ adv )
    return;
candidates [v] = adv;
auto it = candidates . find (v);
it++;
while ( it != candidates.end() && it -> second <= adv )
{   auto temp = it;
    it ++ ;
    candidates . erase (temp);
}
```



Remove unwanted key, value pair

longest common subsequence






```
int lcs ( string str1 , string str2 , int idx1 , int idx2 )
{   if (idx1 == 0 or idx2 == 0)
    {  return 0;
    }  if(dp[idx1][idx2] != -1){return dp[idx1][idx2];}
    if ( str1[idx1 - 1] == str2[idx2 - 1] )
```

```
if ( str1[idx1 - 1] == str2[idx2 - 1] )
{ return dp[idx1][idx2] = lcs( str1, str2, idx1-1, idx2 - 1) + 1;
}
else
return dp[idx1][idx2] = max( lcs( str1, str2, idx1-1, idx2), lcs( str1, str2,
                                                        idx1, idx2 - 1 ));
}
```

Tabulation

```
vector< vector <int>> dp (n+1 , vector<int> (m+1, -1));

For ( i=0; i<n; i++ )
{  For ( j=0; j<m; j++ )
 {
     if ( i==0 or j==0 )
       dp[i][j] = 0;
 }
}

For ( i=1; i<n; i++ )
{ For ( j=1; j<m; j++ )
{
    if ( str1[i-1] == str2[j-1] )
    { dp[i][j] = 1 + dp[i-1][j-1];     (dp[-1][-1] X)
    }
     else
     dp[i][j] = max ( dp[i-1][j], dp[i][j-1] );
}
```

idx1 → abc

idx2 → abc

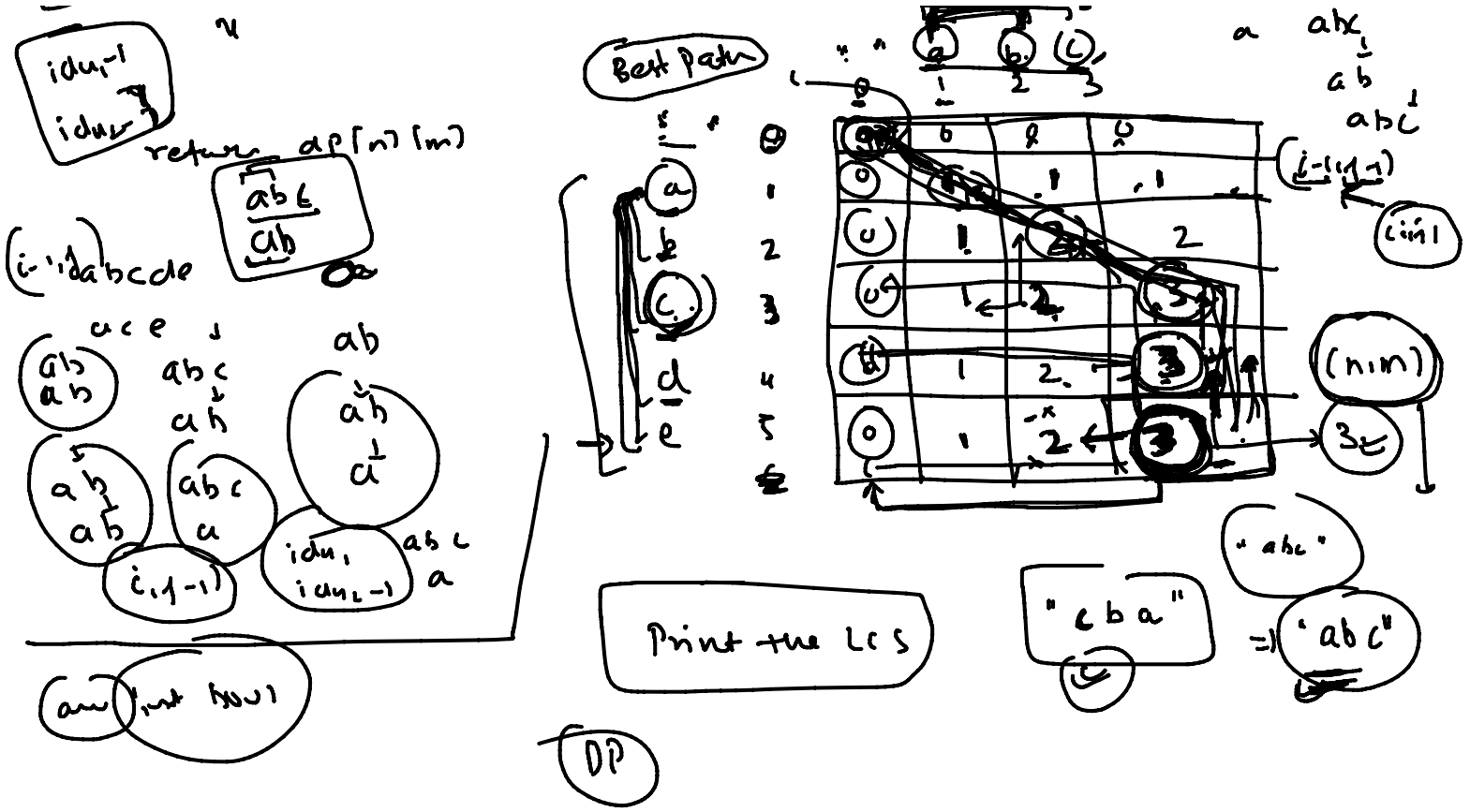
idx1-1

Best Path

ab

a   abc

ab




```
String str = " ";

i=n; j=m.

while (i>0 and j>0)
{   if ( s1[i-1] == s2[j-1] )
    { str.push_back( s1[i-1] ); i--; j--;
    }
    else if ( dp[i-1][j] > dp[i][j-1] )
        i--;
    else
```

Printing the LCS