Class 90

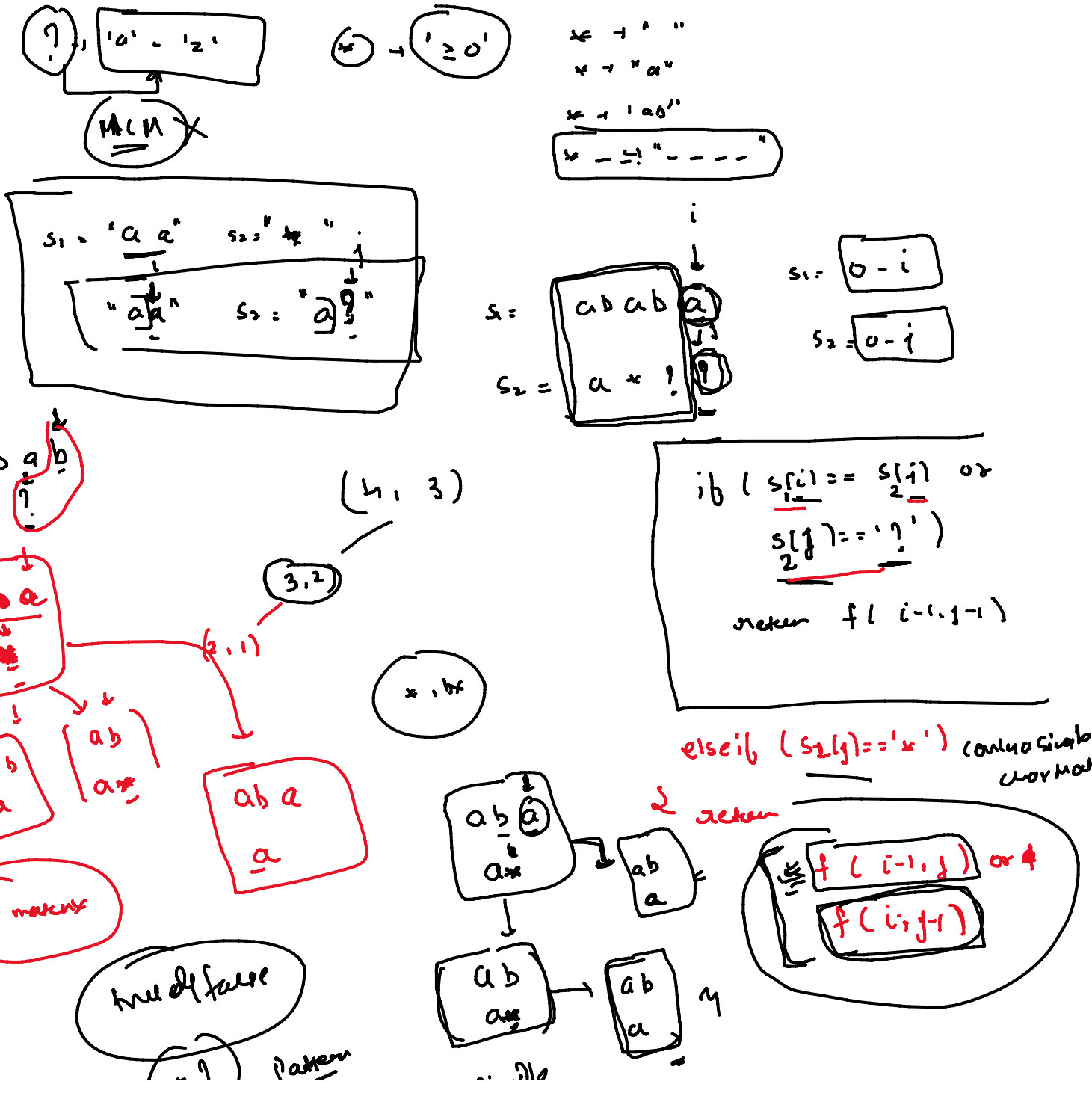
20 May 2023 10:07

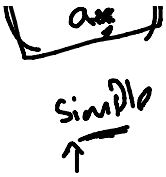
Class 90

Dynamic Programming

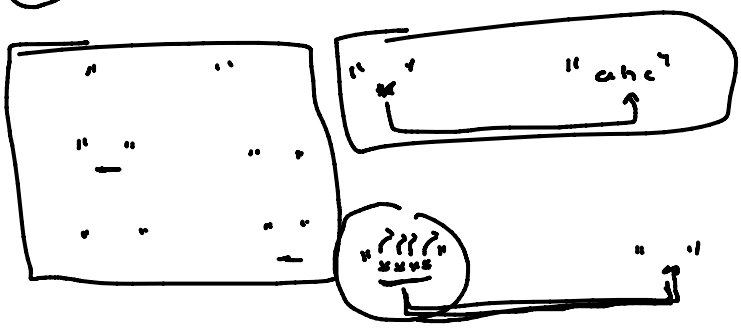
s_1 = abcd

s_2 = a * ?






```
bool solve ( string s1, string s2, int i, int j )
{ if (i<0 and j<0)
  { return true;
  }
  if (i<0 and j>=0)
    return false;
  if (j<0 and i>=0)
  { return isAllStar(s1, i);
  }
  if (dp[i][j]!=-1) return dp[i][j];
  if ( s1[i]==s2[j] or s1[i]=='?' )
  { dp[i][j]=
    return solve ( s1, s2, i-1, j-1);
  }
  else if ( s1[i]=='*' )
  { dp[i][j]=
    return solve ( s1, s2, i, j-1) or solve (s1, s2, i-1, j);
  }
  dp[i][j]=
  return false
}
```



(0 - i)

s2 a b c
s1 = a

Tabulation

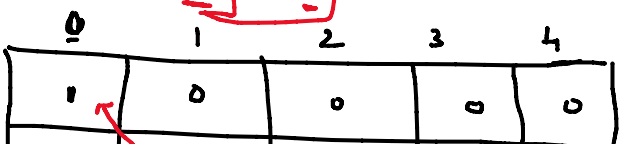
bool
dp(n+1, vector<bool> (m+1, false));

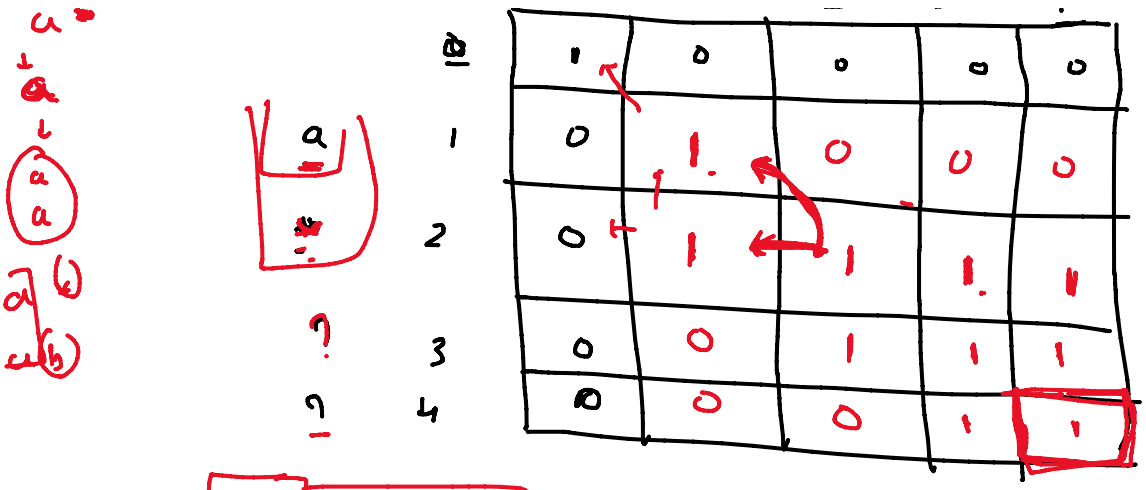
```
bool
Vector<vector<bool>> dp(n+1, vector<bool>(m+1, false));
```

$S_1 = n$
$S_2 = m$

```
dp[0][0] = 1;

for(j = 1; j<=m; j++)
{ dp[0][j] = false;
}

for(i=1; i<=n; i++)
{ dp[i][0] = isAllStore(s1, i);
}

for(i=1; i<=n; i++)
{ for(j=1; j<=m; j++)
  { if(s1[i-1] == s2[j-1] || s1[i-1] == '?')
    { dp[i][j] = dp[i-1][j-1];
    }
    else if(s1[i-1] == '*')
      dp[i][j] = dp[i-1][j] || dp[i][j-1];
    else
    { dp[i][j] = 0
    }
  }
}

return dp[n][m];
```

a b c d





10 10 ? → Space optimize

0 1 1 2 3 5 8 13 - -

20 10 ?

a x ? ?

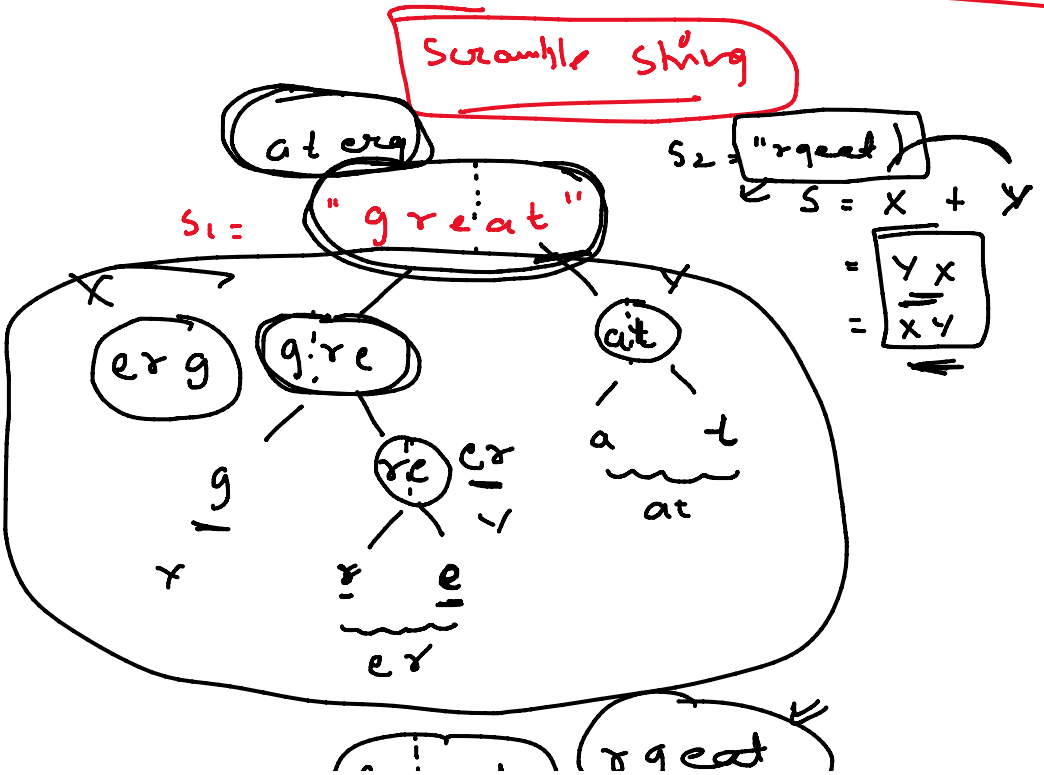
a b c d

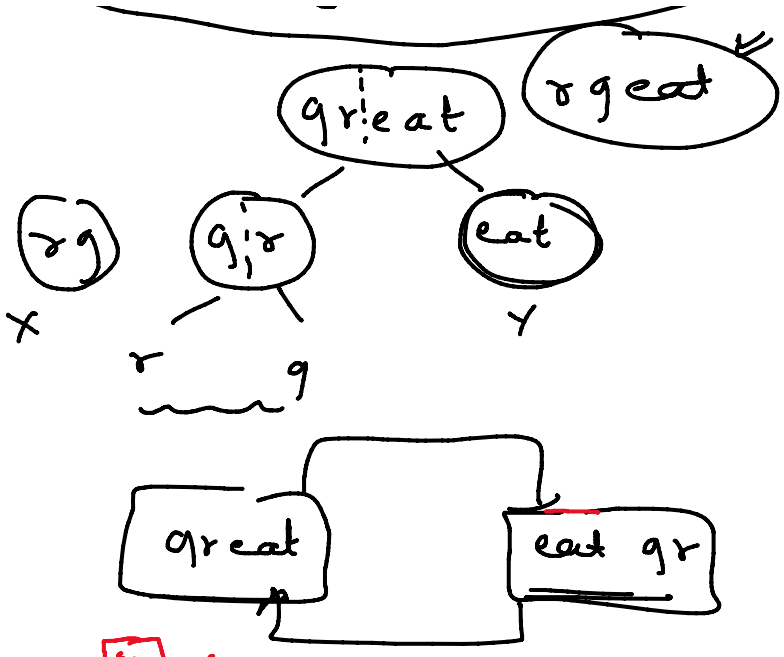
```
vector<int> prev;
vector<int> cur;
vector<int> temp;
swap(prev,cur);
```

~~temp~~

```
temp = prev;
prev = cur;
cur = temp;
```

Scramble string

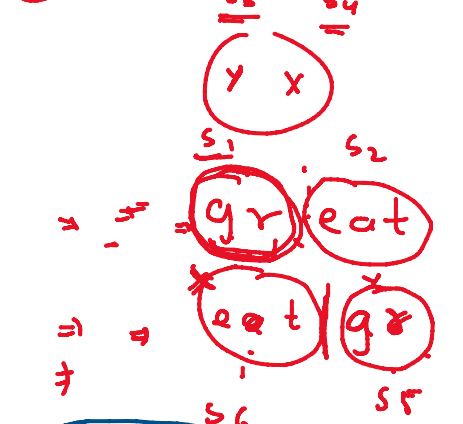




s_1 s_2

s_3 s_4

isScramble (s_1 , s_3) and isScramble (s_2 , s_4)



$x + y \Rightarrow x + y$

$y + x \Rightarrow x + y$

map < pair < string , string > , bool > ma

return [isScramble (s_1 , s_3) and isScramble (s_2 , s_4)] or

= isScramble (s_1 , s_5) and isScramble (s_2 , s_6)

map

map < string , bool > ma ;

a

bool solve (string s_1 , string s_2)

{ if (s_1 . size ( ) == 1)

abc abc

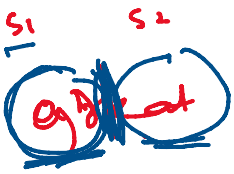
ma . count ({ s_1 , s_2 })

```
if (s1.size() == 1)
{ return s1 == s2
}
if (s1 == s2)
  return true;
int n = s1.size();
for (i=1; i<n; i++)
{
if ((Solve (s1.substr(0,i), s2.substr(0,i)) and
  Solve (s1.substr(i), s2.substr(i))) or
 (Solve (s1.substr(0,i), s2.substr(n-i)) and Solve
  (s1.substr(i), s2.substr(0,n-i)))
}
return true;
}
```

map.count({s1, s2})

string str3 = s1 + ' ' + s2

if (map.count(str3)) return map[str3]



i=2

s1 = abcd

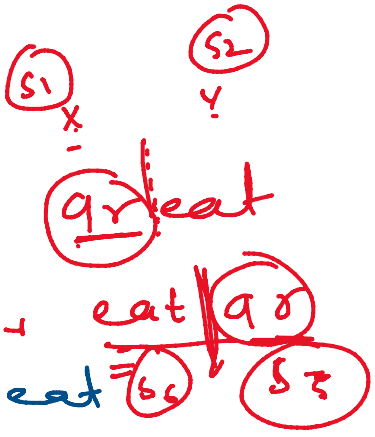
s1.substr(i)

map[{s1, s2}] = true/false

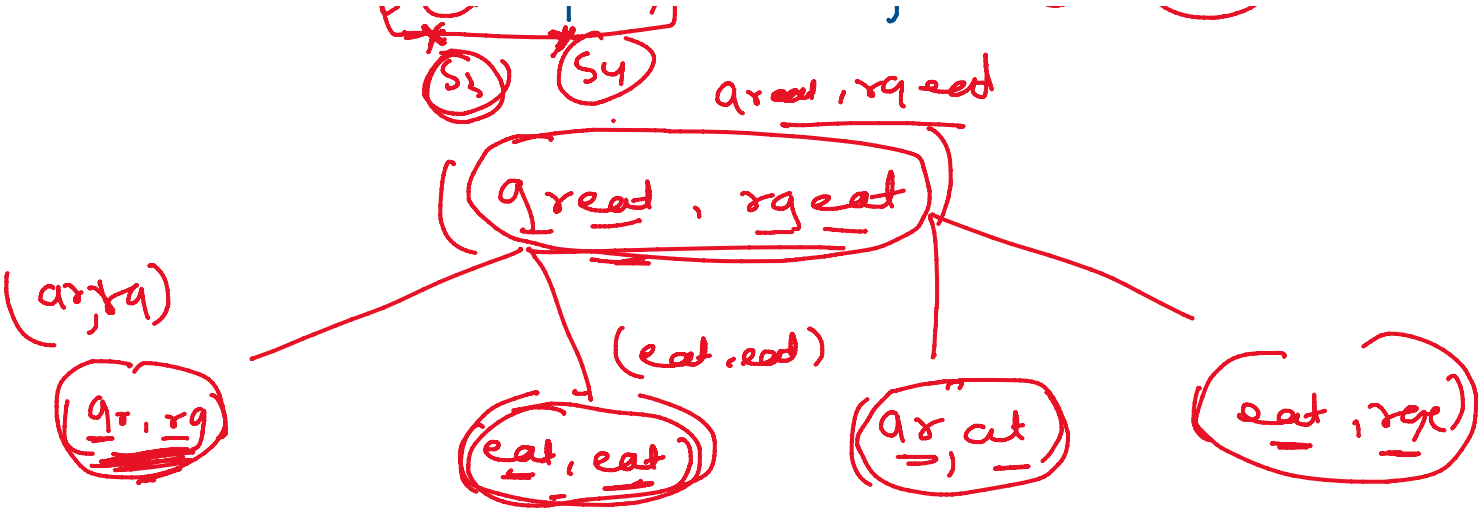
map[str3] =

return false

map[str3]



agreat, rgeat


S1
S4
qread , rqeat
qread , rqeat
(qr,rq)
qr , rq
(eat, eat)
eat , eat
qr , at
eat , rqe