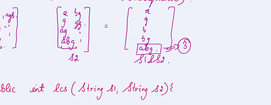
Lecture 28
Monday, 1 June 2020 4:55 PM

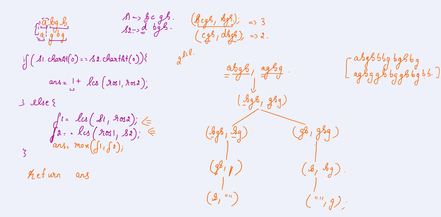
Dynamic Programming.

① String s1 = "abg8" s2 = "ggb" -> lcs (longest common subsequence).

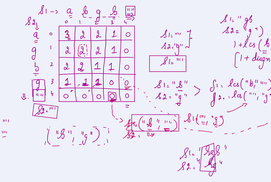


public int lcs(String s1, String s2){

if (s1.charAt(0) == s2.charAt(0)){
ans = 1 + lcs(ros1, ros2);
} else {
f1 = lcs(s1, ros2);
f2 = lcs(ros1, s2);
ans = max(f1, f2);
}
return ans



}

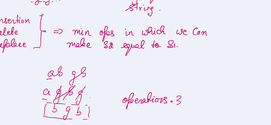


o(s1.length, s2.length);

② edit distance

s1 = "abg8"; s2 = "ggbg"; make s2 string equal to s1 string.

insertion, delete, replace] => min ops in which we can make s2 equal to s1.

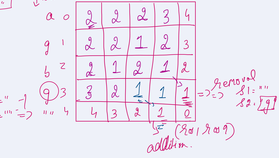


operations = 3

abg8 ggbg => abg8 ggbg => 2 steps
2 ops

s1 => abcd
s2 => "" 4 ops

public int ed(String s1, String s2){
if (s1.length() == 0)
return s2.length();
if (s2.length() == 0) return s1.length();
ros1 = s1.substring(1);
ros2 = s2.substring(1);
if (c1 == c2){
ans = ed(ros1, ros2);
} else {
f1 = ed(ros1, ros2) // replacement
f2 = ed(ros1, s2) // remove
f3 = ed(s1, s2) // add insertion
ans = min(f1, f2, f3) + 1;
}
}



s1 = "g"
s2 = "g"

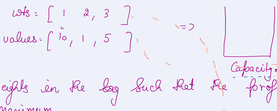
s1: "g"
s2: "g"

s1 = "abg8"
s2 = "ggbg"

Resume at 8:35pm.

0/1 knapsack problem.

wts = [1, 2, 3]
values = [10, 1, 5]



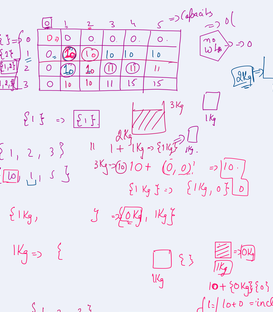
Capacity = 5

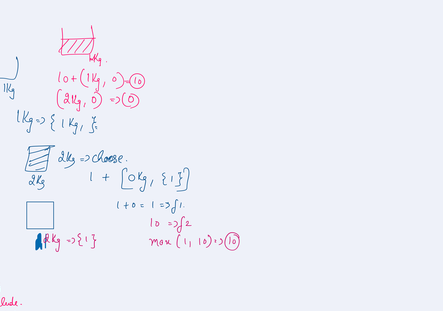
weights in the bag such that the profit is maximum.

1kg 2kg => 3kg <= 5kg | 1kg + 3kg => 4kg <= 5kg | 2kg + 3kg = 5kg
10 + 1 => 11 value | 10 + 5 = 15 | 1 + 5 = 6 => 6.

maximum value.

public int knapsack(int[] wts, int[] values, int Capacity, int index){
if (index == wts.length || Capacity <= 0){
return 0;
}
int ans = 0;
int include = 0;
int exclude = 0;
if (wts[index] < Capacity){
include = values[index] + knapsack(wts, values, Capacity - wts[index], index+1);
exclude = (wts, values, Capacity, index+1);
} else {
exclude = knapsack(wts, values, Capacity, index+1);
}
}
ans = max(include, exclude);
return ans;





=> DP => Google, Amazon.

next session is tomorrow -> 7:00pm

0/1