I think the correct answer to number 6 on the dynamic programming section isn't listed.

Isn't the correct answer "x+1 and items.size"?

Here's the question I'm referring to:

6. Consider memoizing this function:
   function g(x,items,y)
   {
     if (x == 0) return 1;                 //first base case
     if (x < 0) return 0;                 //second base case
     if (y == items.size) return 0;       //third base case
     return minimum(g(x-items[y],items,y),g(x,items,y+1));
   }

Removing all possible base cases, what would be the memoization table's largest index/indices?
Assume no knowledge of the values in items.
A. x + 1 and y
B. x + 1 and y + 1
C. x and y
D. x
E. x + 1
F. x and y + 1
G. x - 1
H. x - 2

This applies to #4 as well.

I think some general clarification on what is meant by removing/retaining base cases would be helpful.

Not sure about #4, but I know the answer to #6 is x+1 and y+2 which was not listed (he confirmed this was the answer in class). I am also stuck on as to why the answers to #4 and #6 would be different due to retaining/removing base cases.
Edit: Because the only base case you cannot remove is the second one, since we don't know if the values are positive.

Subject: Re: Dynamic Programming Question no. 6
Posted by AdamJAlred on Thu, 20 Oct 2016 01:25:43 GMT
View Forum Message <> Reply to Message

For 4, items.size would end up being y+1 since we are checking to see if y equals this and since it is probably a 0 based index, we need to do y+1 instead of y.

Subject: Re: Dynamic Programming Question no. 6
Posted by cewrobel on Thu, 20 Oct 2016 01:43:39 GMT
View Forum Message <> Reply to Message

Adam, so why is #4 y+1 and #6 y+2? I'm a little confused on that.

Subject: Re: Dynamic Programming Question no. 6
Posted by sestephens on Thu, 20 Oct 2016 01:55:06 GMT
View Forum Message <> Reply to Message

I'm still confused about a couple of things, so I'll just type out my logic and please let me know where I'm erring.

#4: x changes and y changes, so we need a table with 2 dimensions.
I'm taking "base cases are retained" to mean that
   >if (x == 0) return 1;
   >if (x < 0) return 0;
   >if (y == items.size) return 0;
are actually kept in the memoized function.
x is decreasing from its original value to 0 (or less), but the base cases handle values 0 and fewer, so we only need a table than can handle values from x down to 1 (since the base case handles 0), so the first dimension is x.
Assuming y starts at 0, it increases until it reaches items.size. It's weird to me to include y in the final answer as a dimension, since it starts at 0 and then changes (including x isn't weird, though, since it starts at a maximum and then descreses.) So, I would think that the second dimension needs to be items.size - 1 + 1 (-1 since the base case handles when y == items.size, +1 because y starts at 0). Thus, the second dimension is items.size. (This is synonymous to the problem explained on http://beastie.cs.ua.edu/cs201/dynamic.html)

#5: The first can be removed, since we can make a table entry for it. The second and third cannot, since they represent out-of-bounds conditions.

#6: x changes and y changes, so we need a table with 2 dimensions.
If I'm right about #5, then
   >if (x == 0) return 1;
is removed.
The first dimension is now x+1, since we're removing the 0 base case and it must now be included
in the table.
The second dimension is still items.size.

Am I going wrong somewhere? It just doesn't make sense to me that y be used in the final answer
when it's just an index, but the index on http://beastie.cs.ua.edu/cs201/dynamic.html is not used in
the dimension.

Subject: Re: Dynamic Programming Question no. 6
Posted by rokohler on Thu, 20 Oct 2016 02:31:26 GMT
View Forum Message <> Reply to Message

Base case 3 can be removed as well since you can store 0 at each table location with a y of
items.size

Subject: Re: Dynamic Programming Question no. 6
Posted by sestephens on Thu, 20 Oct 2016 02:36:32 GMT
View Forum Message <> Reply to Message

Good point. Then by the same logic, the second dimension on #6 would be items.size+1

Subject: Re: Dynamic Programming Question no. 6
Posted by asvivian on Thu, 20 Oct 2016 04:27:01 GMT
View Forum Message <> Reply to Message

It seems you are treating items.size as if the line had said "if(y >= items.size) return 0"

I'm not sure items.size is specifically a bound so much as just one specific row of the memoization
table... don't take my word on that though, I am not 100% sure on that.

edit: staring at the problem thinking I may have just been totally off with this

Subject: Re: Dynamic Programming Question no. 6
Posted by rokohler on Fri, 21 Oct 2016 15:20:58 GMT
View Forum Message <> Reply to Message

It ends up acting as a bound because when it hits that base case the function doesn't recur any
further.