CS 374 **Spring 2015**

Homework 6

Nikhil Unni (nunni2)

3.

$$\text{KSHOT}(p, n, k) = K(p, 1, n, k, \emptyset)$$

$$K(p, i, j, k, \text{strat}) = \begin{cases} 0 & \text{if } k = 0 \\ 0 & \text{if } j \leq i \\ \max \begin{cases} \max_{i+1 \leq x \leq j}(p[x] - p[i] + K(p, x+1, j, k-1, \text{strat} \cup (i, x))) \\ K(p, i+1, j, k, \text{strat}) \end{cases} \end{cases}$$

```
if k = 0 or j ≤ i then
    return 0;
end
hi ← −∞;
for i + 1 ≤ x ≤ j do
    temp ← p[x] − p[i] + K(x + 1, j, k − 1);
    if temp > hi then
        hi = temp;
    end
end
temp ← K(i + 1, j, k);
if temp > hi then
    hi = temp;
end
return hi;
```

Algorithm 1: K-shot strategies

So the algorithm starts out with two indices, i and j, pointing to 1 and n respectively, to encompass the entire initial problem. You can see the correct initial call for the problem, at the top : KSHOT(p,n,k) = $K(p, 1, n, k, \emptyset)$. At every stage, the algorithm has a choice to include i in the list of pairs, or move on, where the second index, x, is anything between i+1 and j, inclusive – in this case, the method is recursively called with the correct indices, a value of k-1, and the strategy so far is the same thing appended with this new (b_i, s_i) tuple.

Or, the algorithm can move on an index, by skipping the current i, and moving on to i+1, with the rest of the parameters the same.

If we've already exhausted the total number of strategies so far, or if we've decremented k, k times (or if k == 0), or if we've reached the bound of our indices i and j, then the algorithm returns 0 to show that it is nonoptimal (yet still better than a negative choice).

I've written out the pseudocode for the same function, although it only calculates the value of the winning k-shot (off by a factor of 1000 from the real value) merely to show the evaluation order. To actually get the set of k-shots, keep track of the set with each recursive call, like in the mathematical formalization above.

As for the actual memoization, there are only $O(kn^2)$ unique problems, and they're all of the substrings of the array p, given the current number of k-shots we have left. We can see this by the indices i,j, and k. The memoization structure woud be a 3D array of size nxnxk, where each position represents the value of K(i,j,k), as the max value of k-shots between a given i and j are also dictated by how many k-shots we are permitted at most. Because each problem takes O(1) time to evaluate, just simple addition and subtraction, in total, the time complexity of the algorithm is $O(kn^2)$.