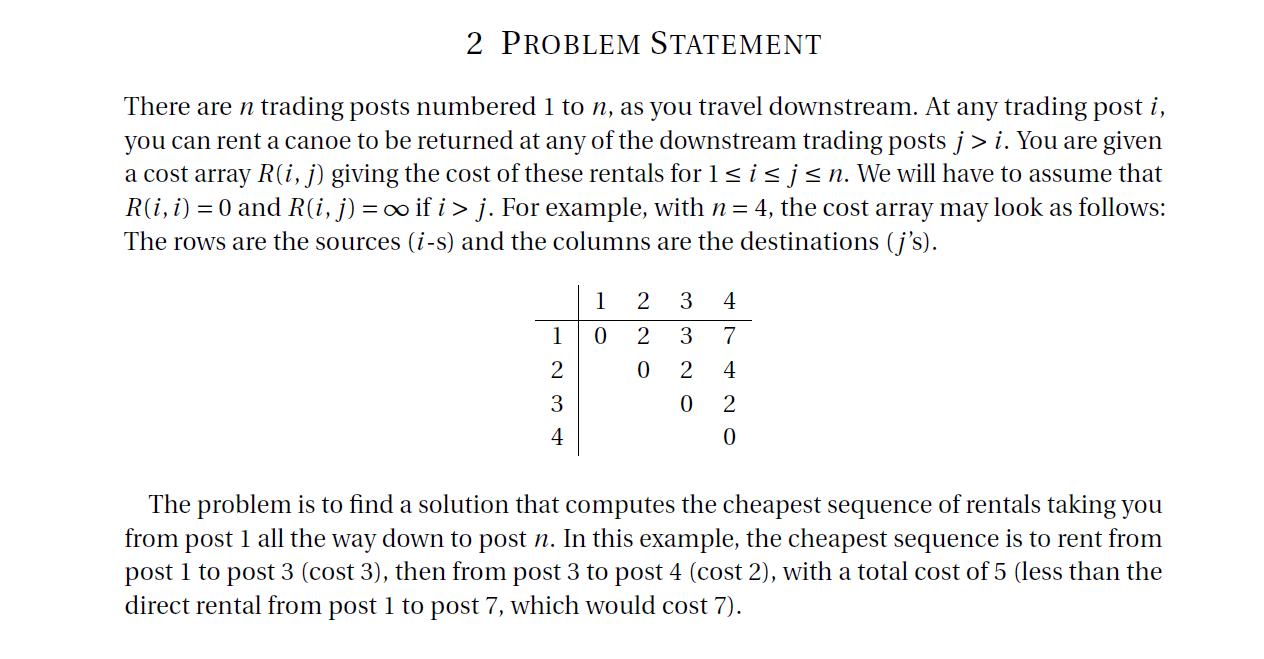
Aaron Chau, Tyler Brent, William Almond

TCSS 343

Spring 2016

Programming Project



Our group project sought to find and analyse the running times of three methods of programming solutions to the problem posted above. The first way we programmed was in a Brute force approach making the most thorough approach that we can considerting every possible solution. Secondly we attacked the problem with the divide and conquer method that divides down the problem into sub problems and solves the problem in that way. And lastly we used the dynamic programming solution that solves the last problem to consider and uses the stored values to solve the problem in the most efficient way of the three.

The labor on this project was divided into three parts, Aaron worked on the Brute Force and Divide and Conquer programming, Will Almond worked mostly on the Dynamic Programming and Documentation, and Tyler Brand worked on the Challenge question and the Testing.

The result of our analysis of each of the runtimes is as follows:

Brute Force:

The brute force algorithm loops through the size of the array twice (making O(n^2) time just for the loops), and compares each element up to k times. We know that k is at most equal to n so the runtime is O(n^3).

Divide and Conquer:

Would have run at a at the usual O(nlogn) time but because it had so many comparisons it ran in O(n^2) time.

Divide and conquer is a faster algorithm that

Dynamic programming:

Because it didn’t have to retrace it’s steps it ran near linear time. But was more like O(nm) time.

C[1]=0

For i=2 to n do

C[i] = C[i-1] + R(i-1, i)

L[i] = i-1

For j=1-2 down to 1 do

If C[j] + R(j, i) < C[i] then

C[i] = C[j] +R(j,i)

L[i]=j

End do

End do

End do

end