CME 2204 Assignment-2

Dynamic Programming and Greedy Approach Report

Ahmet Gürbüz-1509510002

ABSTRACT

In this project we want to compare space complexity vs runtime complexity between Dynamic programming vs greedy Approach. we have done 2 simulation to observe how costly the approaches are in work on small data and large data.

Introduction

The introduction should not include subheadings. In the first simulation, we were expected to establish a dynamic programming structure that would allow a car company to handle with the demands at the least cost. In the second question, we are expected to ensure that this company puts its earnings at the right banks, and we also provide the company with the highest earnings. The way to achieve these can be done by fully understanding the dynamic programming structure. For these two simulations, we will use the tables called matrix and set up a dynamic programming structure on these tables.

Results

For Question (simulation) 1;

Runtime Complexity			
Mounth	**Brute Force**	**Dynamic Programing**	**Greedy**
1	4	4	4
2	16	16	4
3	64	16	4
4	256	16	4
5	1024	16	4
.	.	.	.
.	.	.	.
.	.	.	.
N	**R^N**	**N*R*R**	**N*R**

Space Complexity			
Mounth	**Brute Force**	**Dynamic Programing**	**Greedy**
1	4	4	4
2	16	4	4
3	64	4	4
4	256	4	4
5	1024	4	4
.	.	.	.
.	.	.	.
.	.	.	.
N	**R^N**	**N*R**	**N*R**

R=total demand of N months; N= until N months

For Question (simulation) 2;

Runtime Complexity			
Mounth	**Brute Force**	**Dynamic Programing**	**Greedy**
1	4	4	4
2	16	16	4
3	64	16	4
4	256	16	4
5	1024	16	4
.	.	.	.
.	.	.	.
.	.	.	.
N	**C^N**	**N*C*C**	**N*C**

Space Complexity			
Mounth	**Brute Force**	**Dynamic Programing**	**Greedy**
1	4	4	4
2	16	4	4
3	64	4	4
4	256	4	4
5	1024	4	4
.	.	.	.
.	.	.	.
.	.	.	.
N	**C^N**	**N*C**	**N*C**

C=available bank numbers for this table c=4; N= until N month

Discussion

if we explain the above tables. Using the brute force technique for simulation number 1 is not a serious problem for small numbers. But there is an exponential increase when the number approaches infinity. Since time and cost are the same in brute force, there is a serious loss in both time and memory space. (c ^ N = runtime complexity and space complexity.) so we change our approach. as seen from the table The dynamic programming and the greedy algorithm reveal a much better performance both in runtime and space complexity. There is a difference in accuracy between greedy and dynamic programming. Greedy uses the option to neglect many situations that will be effective when doing operations, when causes us to move away from the desired result as test scenarios become more complex. For this reason, the most accurate and fastest result for these 2 simulations provided us with dynamic programming

Analysis

if we explain how we have achieved these working times, we should analyze my code. If we do run-time analysis for dynamic programming for the first question, we have 3 for loop structures to be seen in our code, and our first loop is the number of months. For the second and third loop, our limiter is the total number of requests. so, the runtime is N * R * R. If we do a runtime analysis of the Greedy approach, we do not need the third loop here because greedy neglects the possibilities we examined on this loop. so, the runtime is N * R. For brute force, the run-time is R ^ N, because it has to be looked at one by one, and each possibility. I understand that every gap in the dynamic programing table is a sub problem. therefore, brute force run time is R^N.

(burada tam anlatmak istediğimi anlatamayacağımdan türkçe yasmak istedim.
Brute force aslında dinamik programlama tablosundaki her ihtimale ve o ihtimallere bakarken oluşacak her ihtimale bakacağı için exponansiyel bir çalışma zamanına sahip. Diğer bir deyişle biz bu ihtimalleri dinamik programlama yaparken tablolaştırıyoruz.)

If we do run-time analysis for dynamic programming for the second question, we have 3 for loop structures again. In the first of these loops, our limiter is again the number of months. (Generally, time is limiting for dynamic programming structures.) For the second and third loop, our limit is the number of banks we can choose. For this reason, working times for this structure are:

-Brute Force =C^N

*-Dynamic Programing = N*C*C*

*-Greedy = N*C*

Field complexity is the same for same methods in 2 questions. Dynamic programming and greedy look at the possibilities but note only the best of them. Brute force writes what it does, the main difference stems from here. So, The area complexity are found as follows by using of loops.

Brute Force =C^N

*-Dynamic Programing = N*C*

*-Greedy = N*C*

Result

As a result of this experiment, while working with large data numbers, we realized that the brute force approach caused a huge degree of runtime and memory loss. We have seen that the Greedy and Dynamic programming approach has a much better runtime and memory loss.
The difference between the Greedy algorithm and the dynamic programming algorithm emerges from accuracy. If we are doing a flexible job in this regard, we can choose greedy to save run time.

References

1. Prof. Dr. Y. İlker Topcu & Assoc. Prof. Dr. Özgür Kabak , OPERATIONS RESEARCH II LECTURE NOTES, 103–116, DOI: https://web.itu.edu.tr/topcuil/ya/END332E.pdf (2019).