**Advanced Algorithms (COMP 257) Project Proposal**

**Simran Nirajkumar Shah**

**GitHub Link:**

**Problem 1:**

You have a certain amount of money available to fund grants. You receive some number of funding proposals, which each have a quality rating and an amount of funding requested. You want to fund a subset of grants that maximize the sum of their quality ratings, without going over your budget. Let’s take an example as follow:

|  |  |  |
| --- | --- | --- |
| Total Funds: $10,000 | | |
| Funding Proposals | Quality Rating | Funding Request |
| FP1 | 120 | $1,500 |
| FP2 | 300 | $5,000 |
| FP3 | 400 | $5,000 |
| FP4 | 500 | $2,000 |
| FP5 | 1000 | $1,000 |
| FP6 | 500 | $3,000 |
| FP7 | 800 | $4,000 |

Here we must find the Maximum Quality rating from the matching funding proposal, but we cannot go above the Total Funds: $10,000

The answer for all the algorithms is as below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Total Funds: $10,000 | | | | |
| Funding Proposals | | | Quality Rating | Funding Request |
|  | FP5 |  | 1000 | $1,000 |
|  | FP4 |  | 500 | $2,000 |
|  | FP7 |  | 800 | $4,000 |
|  | FP6 |  | 500 | $3,000 |
| FP1 | | | 120 | $1,500 |
| FP3 | | | 400 | $5,000 |
| FP2 | | | 300 | $5,000 |

**So, from above answer would be:**

Total Funds Distributed = FP5 + FP4 + FP7 + FP6

= $1000 + $2000 + $4000 + $3000

= $10,000

Maximum Quality Rating = 1000 + 500 + 800 + 500 = 2800

**Types of Algorithms:**

1. Brute Force Algorithm
2. Greedy Algorithm
3. Dynamic Algorithm

**Test-Case:**

**Case 1**

Quality rating = [1000, 500, 800, 500, 120, 400, 300]

Funding Request = [1000, 2000, 4000, 3000, 1500, 5000, 5000]

Total Funds = 10000

**Case 2**

Quality Rating = [10, 50, 80, 50, 12, 40, 30, 40, 20, 100, 1001, 200, 2, 1]

Funding Request = [1, 2, 4, 3, 15, 5, 5, 1, 1, 2, 2, 3, 2, 1]

Total Funds = 60

**Case 3**

Quality Rating = [1, 3, 0, 500, 120, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 1000, 2000]

Funding Request = [0, 2, 4, 3, 15, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2]

Total Funds = 60

**Case 4**

Quality Rating = [50, 60, 7009, 54500, 13420, 43400, 33400]

Funding Request = [100, 200, 400, 300, 150, 500, 500]

Total Funds = -100

**Case 5**

Quality Rating = [323, 554, 434]

Funding Request = [10, 2, 10]

Total Funds = 2

**Test Case and their Expected value:**

**Table

Description automatically generated**

**Brute Force Algorithm:**

***Input:***

**Case 1**

QualityRating = [1000, 500, 800, 500, 120, 400, 300]

FundingRequest = [1000, 2000, 4000, 3000, 1500, 5000, 5000]

TotalFunds = 10000

ListLen = len(QualityRating)

**Case 2**

QualityRating = [10, 50, 80, 50, 12, 40, 30, 40, 20, 100, 1001, 200, 2, 1]

FundingRequest = [1, 2, 4, 3, 15, 5, 5, 1, 1, 2, 2, 3, 2, 1]

TotalFunds = 60

ListLen = len(QualityRating)

**Case 3**

QualityRating = [1, 3, 0, 500, 120, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 1000, 2000]

FundingRequest = [0, 2, 4, 3, 15, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2]

TotalFunds = 60

ListLen = len(QualityRating)

**Case 4**

QualityRating = [50, 60, 7009, 54500, 13420, 43400, 33400]

FundingRequest = [100, 200, 400, 300, 150, 500, 500]

TotalFunds = -100

ListLen = len(QualityRating)

**Case 5**

QualityRating = [323, 554, 434]

FundingRequest = [10, 2, 10]

TotalFunds = 2

ListLen = len(QualityRating)

***Output:***

Maximum Quantity: 2800 and Run time = 0.0

Maximum Quantity: 1636 and Run time = 0.014516115188598633

Maximum Quantity: 47504 and Run time = 0.018151521682739258

Maximum Quantity: 0 and Run time = 0.0

Maximum Quantity: 554 and Run time = 0.0

***Brute Force Rule:***

In this algorithm, all the possible values are matched with each other and after the overall match up conclusion is shown on the screen. Here I have used Recursion without Memoization to make all possible values. With different types of the test cases (5).

***Pseudocode*:**

def MaxQualityRating(FundingRequest, QualityRating, LengthList, TotalFunds):

check if LengthList == 0 | TotalFunds == 0:

return NULL

if FundingRequest for last taken proposal is (FundingRequest[LengthList - 1]) > TotalFunds:

return MaxQualityRating(FundingRequest, QualityRating, ListLen - 1, TotalFunds)

if Funds are reduced then ReduceFundingRequest = MaxQualityRating(FundingRequest, QualityRating,

LengthList - 1, TotalFunds - FundingRequest[ListLen - 1])

if funds are not reduced then NotReduceFundingRequest = MaxQualityRating(FundingRequest,

QualityRating, LengthList - 1, TotalFunds)

return max(QualityRating[ListLen - 1] + ReduceFundingRequest, NotReduceFundingRequest)

***Code:***

from time import time

def BruteForce(FundingRequest, QualityRating, ListLen, TotalFunds):

if ListLen <= 0 or TotalFunds <= 0:

return 0

if FundingRequest[ListLen - 1] > TotalFunds:

return BruteForce(FundingRequest, QualityRating, ListLen - 1, TotalFunds)

return max(QualityRating[ListLen - 1] + BruteForce(FundingRequest, QualityRating, ListLen - 1,

TotalFunds - FundingRequest[ListLen - 1]),

BruteForce(FundingRequest, QualityRating, ListLen - 1, TotalFunds))

if \_\_name\_\_ == '\_\_main\_\_':

init = time()

QualityRating = [1000, 500, 800, 500, 120, 400, 300]

FundingRequest = [1000, 2000, 4000, 3000, 1500, 5000, 5000]

TotalFunds = 10000

ListLen = len(QualityRating)

print("Maximum Quantity: " + str(BruteForce(FundingRequest, QualityRating, ListLen, TotalFunds)) + " and Run time = " + str(time()- init))

init = time()

QualityRating = [10, 50, 80, 50, 12, 40, 30, 40, 20, 100, 1001, 200, 2, 1]

FundingRequest = [1, 2, 4, 3, 15, 5, 5, 1, 1, 2, 2, 3, 2, 1]

TotalFunds = 60

ListLen = len(QualityRating)

print("Maximum Quantity: " + str(BruteForce(FundingRequest, QualityRating, ListLen, TotalFunds)) + " and Run time = " + str(time()- init))

init = time()

QualityRating = [1, 3, 0, 500, 120, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 1000, 2000]

FundingRequest = [0, 2, 4, 3, 15, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2]

TotalFunds = 60

ListLen = len(QualityRating)

print("Maximum Quantity: " + str(BruteForce(FundingRequest, QualityRating, ListLen, TotalFunds)) + " and Run time = " + str(time()- init))

init = time()

QualityRating = [50, 60, 7009, 54500, 13420, 43400, 33400]

FundingRequest = [100, 200, 400, 300, 150, 500, 500]

TotalFunds = -100

ListLen = len(QualityRating)

print("Maximum Quantity: " + str(BruteForce(FundingRequest, QualityRating, ListLen, TotalFunds)) + " and Run time = " + str(time()- init))

init = time()

QualityRating = [323, 554, 434]

FundingRequest = [10, 2, 10]

TotalFunds = 2

ListLen = len(QualityRating)

print("Maximum Quantity: " + str(BruteForce(FundingRequest, QualityRating, ListLen, TotalFunds)) + " and Run time = " + str(time()- init))

***Time Complexity:***

The time complexity here depends on the number of elements in the list. As it is evaluating all the possible sets of the solution. Time complexity = O(2^n).

***Run time: (Depends on the length of elements in the array (Funding Array))***

Case 1: 0.0

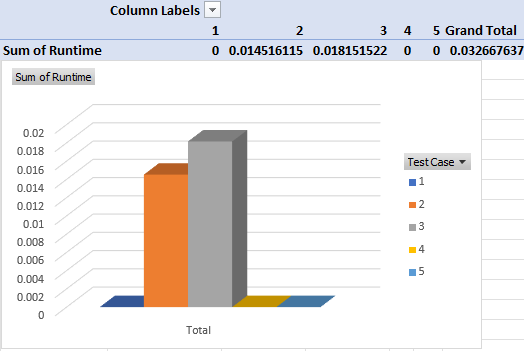
Case 2: 0.015805959701538086 (Length Size more of array passing)

Case 3: 0.014644145965576172 (Length Size more of array passing)

Case 4: 0.0

Case 5: 0.0

***Bar Graph:***

****

**Greedy Algorithm:**

***Input*:**

**Case 1**

TotalFunds = 10000

FundingArray = [Greedy(1000, 1000), Greedy(500, 2000), Greedy(800, 4000), Greedy(500, 3000), Greedy(120, 1500),Greedy(400, 5000), Greedy(300, 5000)]

LengthList = len(FundingArray)

**Case 2**

TotalFunds = 60

FundingArray = [Greedy(10, 1), Greedy(50, 2), Greedy(80, 4), Greedy(50, 3), Greedy(12, 15),

Greedy(40, 5), Greedy(30, 5), Greedy(40, 1), Greedy(20, 1), Greedy(100, 2),

Greedy(1001, 2), Greedy(200, 3), Greedy(2, 2), Greedy(1, 1)]

LengthList = len(FundingArray)

**Case 3**

TotalFunds = 60

FundingArray = [Greedy(1, 0), Greedy(3, 2), Greedy(0, 4), Greedy(500, 3), Greedy(120, 15),

Greedy(2000, 2), Greedy(3000, 3), Greedy(4000, 4), Greedy(5000, 5), Greedy(6000, 6),

Greedy(7000, 7), Greedy(8000, 8), Greedy(9000, 9), Greedy(1000, 1), Greedy(2000, 2)]

LengthList = len(FundingArray)

**Case 4**

TotalFunds = -100

FundingArray = [Greedy(50, 100), Greedy(60, 200), Greedy(7009, 400), Greedy(54500, 300), Greedy(13420, 150),

Greedy(43400, 500), Greedy(33400, 500)]

LengthList = len(FundingArray)

**Case 5**

TotalFunds = 2

FundingArray = [Greedy(323, 10), Greedy(554, 2), Greedy(434, 10)]

LengthList = len(FundingArray)

***Output:***

Maximum Quantity: 2800.0 and Run time = 0.0

Maximum Quantity: 1636 and Run time = 0.0

Maximum Quantity: 47580.0 and Run time = 0.0

Maximum Quantity: 0 and Run time = 0.0

Maximum Quantity: 554.0 and Run time = 0.0

***Greedy Rule:*** Calculating Quality Rating per Dollar for each proposal and sorting them in descending order. In this algorithm, at every stage, the input differs, and the answers too.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | Total Funds: $10,000 | | |  |
| Funding Proposals | | | Quality Rating | Funding Request | per $ |  |
|  | FP5 |  | 1000 | $1,000 |  | 1.00 |
|  | FP4 |  | 500 | $2,000 |  | 0.25 |
|  | FP7 |  | 800 | $4,000 |  | 0.20 |
|  | FP6 |  | 500 | $3,000 |  | 0.17 |
| FP1 | | | 120 | $1,500 |  | 0.08 |
| FP3 | | | 400 | $5,000 |  | 0.08 |
| FP2 | | | 300 | $5,000 |  | 0.06 |

***Pseudocode*:**

def MaxQualityRating(Proposal, FundingRequest, QualityRating, LengthList, TotalFunds):

QualityPerDollarList = []

Calculate Quality parameter per Dollar For i = 1:n:

QualityPerDollarList.append(QualityRating[i] / FundingRequest[i])

while TotalFunds > 0 and proposal is not checked PC= No:

Search QPD1 (High) and Proposal checked PC= No

If TotalFunds > = FundingRequest:

SumQualityParameter = SumQualityParameter + QualityRating

TotalFunds = TotalFunds – FundingRequest

Proposal checked PC= Yes

return SumQualityParameter

***Code:***

from time import time

class Greedy:

def \_\_init\_\_(self, FundingRequest, QualityRating):

self.FundingRequest = FundingRequest

self.QualityRating = QualityRating

class Greedy:

def \_\_init\_\_(self, QualityRating, FundingRequest):

self.FundingRequest = FundingRequest

self.QualityRating = QualityRating

def GreedyAlgo(TotalFunds, FundingArray, LengthList):

if TotalFunds <= 0:

return 0

FundingArray.sort(key=lambda GAlgo: GAlgo.QualityRating / GAlgo.FundingRequest if GAlgo.QualityRating > 0 else 1,

reverse=True)

SumQualityParameter = 0

count = 0

while count < LengthList:

if FundingArray[count].FundingRequest <= TotalFunds:

TotalFunds = TotalFunds - FundingArray[count].FundingRequest

SumQualityParameter = SumQualityParameter + FundingArray[count].QualityRating

count += 1

return SumQualityParameter

if \_\_name\_\_ == '\_\_main\_\_':

init = time()

TotalFunds = 10000

FundingArray = [Greedy(1000, 1000), Greedy(500, 2000), Greedy(800, 4000), Greedy(500, 3000), Greedy(120, 1500),

Greedy(400, 5000), Greedy(300, 5000)]

LengthList = len(FundingArray)

print("Maximum Quantity: " + str(GreedyAlgo(TotalFunds, FundingArray, LengthList)) + " and Run time = " + str(time() - init))

init = time()

TotalFunds = 60

FundingArray = [Greedy(10, 1), Greedy(50, 2), Greedy(80, 4), Greedy(50, 3), Greedy(12, 15),

Greedy(40, 5), Greedy(30, 5), Greedy(40, 1), Greedy(20, 1), Greedy(100, 2),

Greedy(1001, 2), Greedy(200, 3), Greedy(2, 2), Greedy(1, 1)]

LengthList = len(FundingArray)

print("Maximum Quantity: " + str(GreedyAlgo(TotalFunds, FundingArray, LengthList)) + " and Run time = " + str(time() - init))

init = time()

TotalFunds = 60

FundingArray = [Greedy(-1, 0), Greedy(-3, 2), Greedy(0, 4), Greedy(500, 3), Greedy(120, 15),

Greedy(2000, 2), Greedy(3000, 3), Greedy(4000, 4), Greedy(5000, 5), Greedy(6000, 6),

Greedy(7000, 7), Greedy(8000, 8), Greedy(9000, 9), Greedy(1000, 1), Greedy(2000, 2)]

LengthList = len(FundingArray)

print("Maximum Quantity: " + str(GreedyAlgo(TotalFunds, FundingArray, LengthList)) + " and Run time = " + str(time() - init))

init = time()

TotalFunds = -100

FundingArray = [Greedy(50, 100), Greedy(60, 200), Greedy(7009, 400), Greedy(54500, 300), Greedy(13420, 150),

Greedy(43400, 500), Greedy(33400, 500)]

LengthList = len(FundingArray)

print("Maximum Quantity: " + str(GreedyAlgo(TotalFunds, FundingArray, LengthList)) + " and Run time = " + str(time() - init))

init = time()

TotalFunds = 2

FundingArray = [Greedy(323, 10), Greedy(554, 2), Greedy(434, 10)]

LengthList = len(FundingArray)

print("Maximum Quantity: " + str(GreedyAlgo(TotalFunds, FundingArray, LengthList)) + " and Run time = " + str(time() - init))

***Time Complexity:***

Here time complexity is O(n) as we have a loop continuously working.

***Run time:***

Case 1: 0.0

Case 2: 0.0

Case 3: 0.0

Case 4: 0.0

Case 5: 0.0

Note: It depends upon the upcoming factor so if the factor is less than that, the Total funds can be calculated easily.

***Bar graph:***

**A picture containing table

Description automatically generated**

**Dynamic Algorithm:**

***Input:***

**Case 1**

QualityRating = [1000, 500, 800, 500, 120, 400, 300]

FundingRequest = [1000, 2000, 4000, 3000, 1500, 5000, 5000]

TotalFunds = 10000

ListLen = len(QualityRating)

**Case 2**

QualityRating = [10, 50, 80, 50, 12, 40, 30, 40, 20, 100, 1001, 200, 2, 1]

FundingRequest = [1, 2, 4, 3, 15, 5, 5, 1, 1, 2, 2, 3, 2, 1]

TotalFunds = 60

ListLen = len(QualityRating)

**Case 3**

QualityRating = [1, 3, 0, 500, 120, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 1000, 2000]

FundingRequest = [0, 2, 4, 3, 15, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2]

TotalFunds = 60

ListLen = len(QualityRating)

**Case 4**

QualityRating = [50, 60, 7009, 54500, 13420, 43400, 33400]

FundingRequest = [100, 200, 400, 300, 150, 500, 500]

TotalFunds = -100

ListLen = len(QualityRating)

**Case 5**

QualityRating = [323, 554, 434]

FundingRequest = [10, 2, 10]

TotalFunds = 2

ListLen = len(QualityRating)

***Output:***

Maximum Quantity: 2800 and Run time = 0.024263381958007812

Maximum Quantity: 1636 and Run time = 0.0

Maximum Quantity: 47504 and Run time = 0.0010008811950683594

Maximum Quantity: 0 and Run time = 0.0

Maximum Quantity: 554 and Run time = 0.0

***Dynamic Rule:***

Here list is put into 2-dimensional array according to the FundingCapacity and QualityRating because of their Ranges.

***Pseudocode:***

def MaxQualityRating(FundingRequest, QualityRating, LengthList, TotalFunds):

Create a chart for 2 dimensional list chart[i][j] using [[1:TotalFunds + 1] and Lengthlist + 1]

create a for i in range(1:LengthList + 1):

calculate for j in range(1:TotalFunds + 1):

if FundingRequest[i - 1] <= j:

Chart[i][j] = max(QualityRating[i -1 +

Chart[i - 1][j -FundingRequest[i - 1]],

Chart[i - 1][j])

Chart[i][j] = Chart[i - 1][j]

return Chart[LengthList][TotalFunds]

***Code:***

from time import time

def DynamicAlgo(FundingRequest, QualityRating, LengthList, TotalFunds):

if ListLen <= 0 or TotalFunds <= 0:

return 0

Table = [[0 for x in range(0, TotalFunds + 1)]

for x in range(0, LengthList + 1)]

for x in range(0, LengthList + 1):

for z in range(0, TotalFunds + 1):

if x <= 0 or z <= 0:

Table[x][z] = 0

elif FundingRequest[x - 1] <= z:

Table[x][z] = max(QualityRating[x - 1]

+ Table[x - 1][z - FundingRequest[x - 1]],

Table[x - 1][z])

else:

Table[x][z] = Table[x - 1][z]

return Table[LengthList][TotalFunds]

if \_\_name\_\_ == '\_\_main\_\_':

init = time()

QualityRating = [1000, 500, 800, 500, 120, 400, 300]

FundingRequest = [1000, 2000, 4000, 3000, 1500, 5000, 5000]

TotalFunds = 10000

ListLen = len(QualityRating)

print("Maximum Quantity: " + str(DynamicAlgo(FundingRequest, QualityRating, ListLen, TotalFunds)) + " and Run time = " + str(time()- init))

init = time()

QualityRating = [10, 50, 80, 50, 12, 40, 30, 40, 20, 100, 1001, 200, 2, 1]

FundingRequest = [1, 2, 4, 3, 15, 5, 5, 1, 1, 2, 2, 3, 2, 1]

TotalFunds = 60

ListLen = len(QualityRating)

print("Maximum Quantity: " + str(DynamicAlgo(FundingRequest, QualityRating, ListLen, TotalFunds)) + " and Run time = " + str(time()- init))

init = time()

QualityRating = [1, 3, 0, 500, 120, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 1000, 2000]

FundingRequest = [0, 2, 4, 3, 15, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2]

TotalFunds = 60

ListLen = len(QualityRating)

print("Maximum Quantity: " + str(DynamicAlgo(FundingRequest, QualityRating, ListLen, TotalFunds)) + " and Run time = " + str(time()- init))

init = time()

QualityRating = [50, 60, 7009, 54500, 13420, 43400, 33400]

FundingRequest = [100, 200, 400, 300, 150, 500, 500]

TotalFunds = -100

ListLen = len(QualityRating)

print("Maximum Quantity: " + str(DynamicAlgo(FundingRequest, QualityRating, ListLen, TotalFunds)) + " and Run time = " + str(time()- init))

init = time()

QualityRating = [323, 554, 434]

FundingRequest = [10, 2, 10]

TotalFunds = 2

ListLen = len(QualityRating)

print("Maximum Quantity: " + str(DynamicAlgo(FundingRequest, QualityRating, ListLen, TotalFunds)) + " and Run time = " + str(time()- init))

***Time Complexity:***

Here we can calculate time complexity as O(F\*N) depends on Funding Capacity and the number of elements present in the list.

***Run time: (Depends on the ratio of elements in the array)***

Case 1: 0.028481006622314453(Quantity Ratio more)

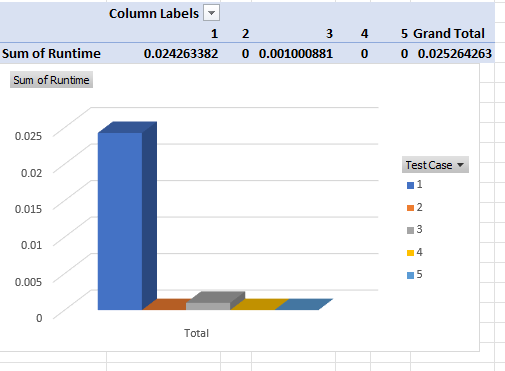
Case 2: 0.0

Case 3: 0.0010645389556884766

Case 4: 0.0

Case 5: 0.0

***Bar graph:***

****

***Recommendation:*** Greedy Algorithm

Here I would like to choose the Greedy Algorithm as it has a smart and optimal solution, and we don’t have to iterate to the whole array for matches. And its Time complexity is also O(N) which is the lowest.

Text, table

Description automatically generated

Shown in Bar Graph:

Chart

Description automatically generated

-------------------------------------------------------------THANK YOU-------------------------------------------------------------