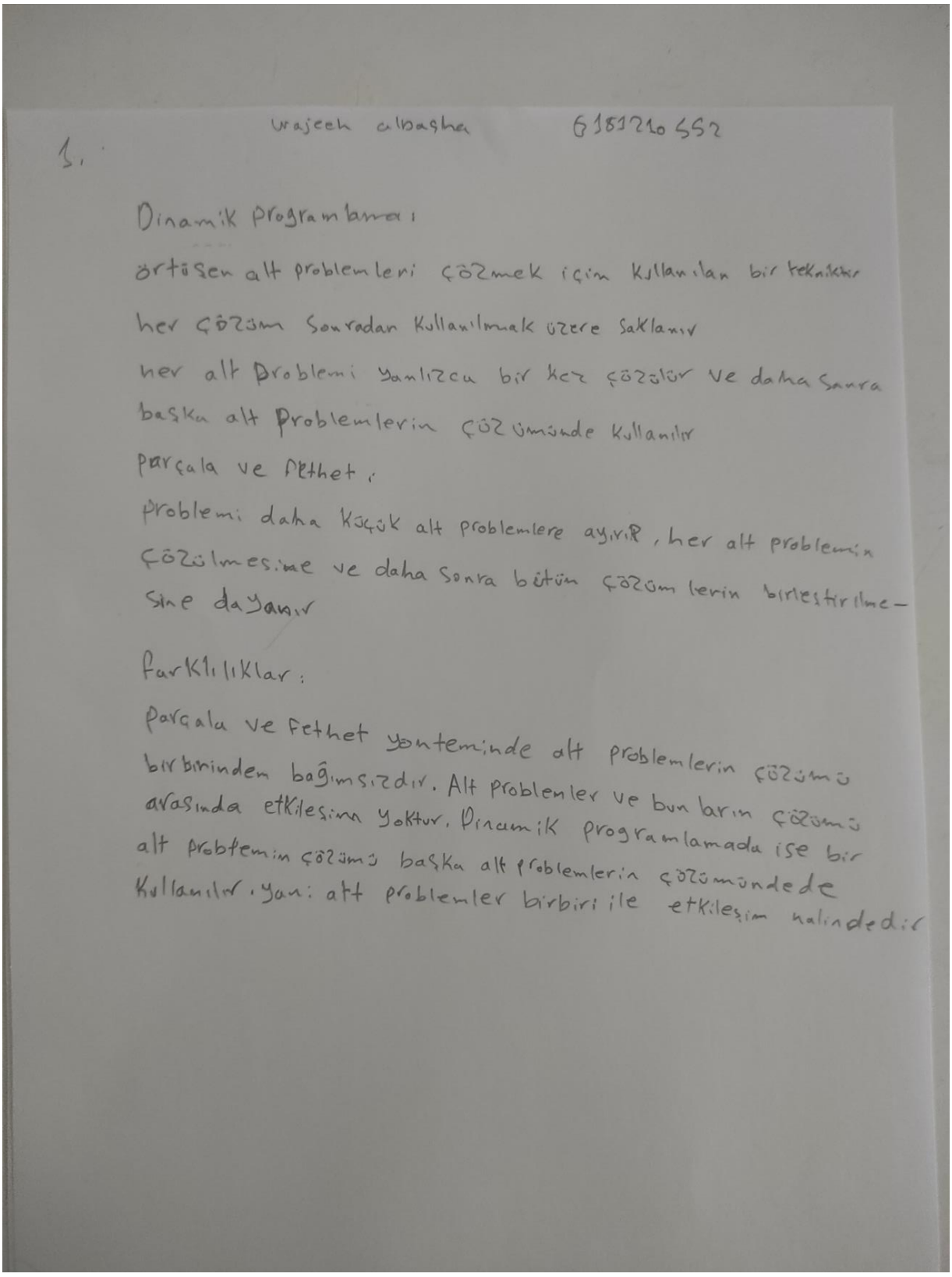
1. Soru :



1.

Vrajeen albasha 6181210552

Dinamik programlama:

örtüsen alt problemleri çözmek için kullanılan bir teknikter

her çözüm sonradan kullanılmak üzere saklanır

her alt problemi yanlızca bir kez çözülür ve daha sonra başka alt problemlerin çözümünde kullanılır

parçala ve fethet:

problemi daha küçük alt problemlere ayırır, her alt problemin çözülmesine ve daha sonra bütün çözümlerin birleştirilmesine dayanır

farklılıklar:

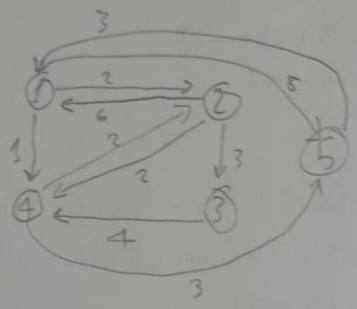
parçala ve fethet yönteminde alt problemlerin çözümü birbirinden bağımsızdır. Alt problemler ve bunların çözümü arasında etkileşim yoktur. Dinamik programlamada ise bir alt problemin çözümü başka alt problemlerin çözümündede kullanılır. yani: alt problemler birbiri ile etkileşim halindedir

2. soru :

2.) 1 2 3 4 5 Wajeeh albasha 0181210552

$$A^0 = \begin{bmatrix} 0 & 2 & \infty & 1 & 8 \\ 6 & 0 & 3 & 2 & \infty \\ \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 0 & 3 \\ 3 & \infty & \infty & \infty & 0 \end{bmatrix}$$

formül: $A^k[i,j] = \min\{A^{k-1}[i,j], A^{k-1}[i,K] + A^{k-1}[k,j]\}$



2. Soru

$$A^1 = \begin{bmatrix} 0 & 2 & \infty & 1 & 8 \\ 6 & 0 & 3 & 2 & 14 \\ \infty & \infty & 0 & 4 & 8 \\ \infty & \infty & 2 & 0 & 3 \\ 3 & 5 & \infty & 4 & 0 \end{bmatrix}$$

$A^1[2,3] = \min\{A^0[2,3], A^0[2,1] + A^0[1,3]\}$
$= \min\{3, 6+8\} = 3$

$A^1[2,4] = \min\{2, 6+1\} = 2$

$A^1[2,5] = \min\{\infty, 6+8\} = 14$

$A^1[3,2] = \min\{\infty, \infty+2\} = \infty$

$A^1[3,4] = 4, \; A^1[3,5] = \infty$

$A^1[4,2] = \infty, \; A^1[4,3] = 2, \; A^1[4,5] = 3$

$$A^2 = \begin{bmatrix} 0 & 2 & 5 & 1 & 8 \\ 6 & 0 & 3 & 2 & 14 \\ \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 0 & 3 \\ 3 & 5 & 8 & 4 & 0 \end{bmatrix}$$

$A^2[1,3] = \min\{A^1[1,3], A^1[1,2] + A^1[2,3]\}$
$= \min\{\infty, 2+3\} = 5$

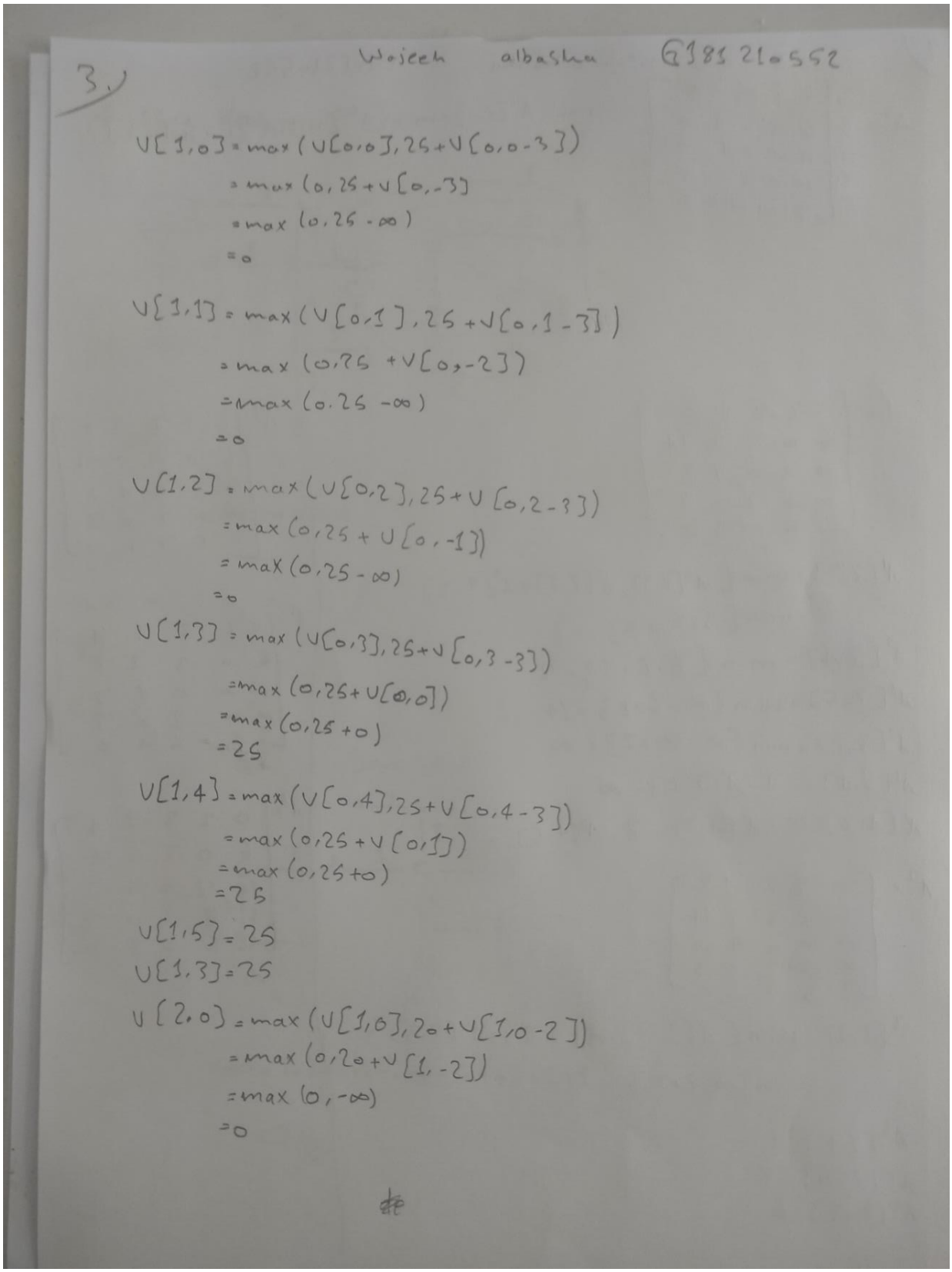
$A^2[1,4] = 1$
$A^2[1,5] = 8$
$A^2[2,1] = 6$

$$A^3 = \begin{bmatrix} 0 & 2 & 5 & 1 & 8 \\ 6 & 0 & 3 & 2 & 14 \\ \infty & \infty & 0 & 4 & 8 \\ \infty & \infty & 2 & 0 & 3 \\ 3 & 5 & 8 & 4 & 0 \end{bmatrix}$$

$$A^4 = \begin{bmatrix} 0 & 2 & 3 & 1 & 4 \\ 6 & 0 & 3 & 2 & 5 \\ \infty & \infty & 0 & 4 & 7 \\ \infty & \infty & 2 & 0 & 3 \\ 3 & 5 & 6 & 4 & 0 \end{bmatrix}$$

$$A^5 = \begin{bmatrix} 0 & 2 & 3 & 1 & 4 \\ 6 & 0 & 3 & 2 & 5 \\ 10 & 12 & 0 & 4 & 7 \\ 6 & 8 & 2 & 0 & 3 \\ 3 & 5 & 6 & 4 & 0 \end{bmatrix}$$

3. soru :



3.) Wojcech albaska 2181210552

$V[1,0] = \max(V[0,0], 25 + V[0,0-3])$
$= \max(0, 25 + V[0,-3])$
$= \max(0, 25 - \infty)$
$= 0$

$V[1,1] = \max(V[0,1], 25 + V[0,1-3])$
$= \max(0, 25 + V[0,-2])$
$= \max(0, 25 - \infty)$
$= 0$

$V[1,2] = \max(V[0,2], 25 + V[0,2-3])$
$= \max(0, 25 + V[0,-1])$
$= \max(0, 25 - \infty)$
$= 0$

$V[1,3] = \max(V[0,3], 25 + V[0,3-3])$
$= \max(0, 25 + V[0,0])$
$= \max(0, 25 + 0)$
$= 25$

$V[1,4] = \max(V[0,4], 25 + V[0,4-3])$
$= \max(0, 25 + V[0,1])$
$= \max(0, 25 + 0)$
$= 25$

$V[1,5] = 25$

$V[1,3] = 25$

$V[2,0] = \max(V[1,0], 20 + V[1,0-2])$
$= \max(0, 20 + V[1,-2])$
$= \max(0, -\infty)$
$= 0$

Wajeeh albasha 6181210552

3. Soru devamı

V[2,1] = 0
V[2,2] = 20
V[2,3] = 20
V[2,4] = 20
V[2,5] = 45
V[2,6] = 45

V[3,0] = max(V[2,0], 15 + V[2,0-1])
= max(0, 15 + V[2,-1])
= max(0, -∞)
= 0

V[3,1] = 15
V[3,2] = 15
V[3,3] = 15 + V[2,2] = 15 + 20 = 35
V[3,4] = 15 + V[2,3] = 15 + 20 = 35
V[3,5] = 15 + V[2,4] = 15 + 20 = 35
V[3,6] = 15 + V[2,5] = 15 + 45 = 60

V[4,0] = max(V[3,0], 40 + V[3,0-4])
= max(0, 40 + V[3,-4])
= max(0, 40 - ∞)
= max(0, -∞)
= 0

V[4,1] = 0
V[4,2] = 0
V[4,3] = 0
V[4,4] = 40
V[4,5] = 40 + V[3,1] = 40 + 15 = 55
V[4,6] = 40 + V[3,2] = 40 + 15 = 55

V[5,0] = max(V[4,0], 50 + V[4,0-5])
= max(0, 50 + V[4,-4])
= max(0, 50 - ∞)
= max(0, -∞)
= 0

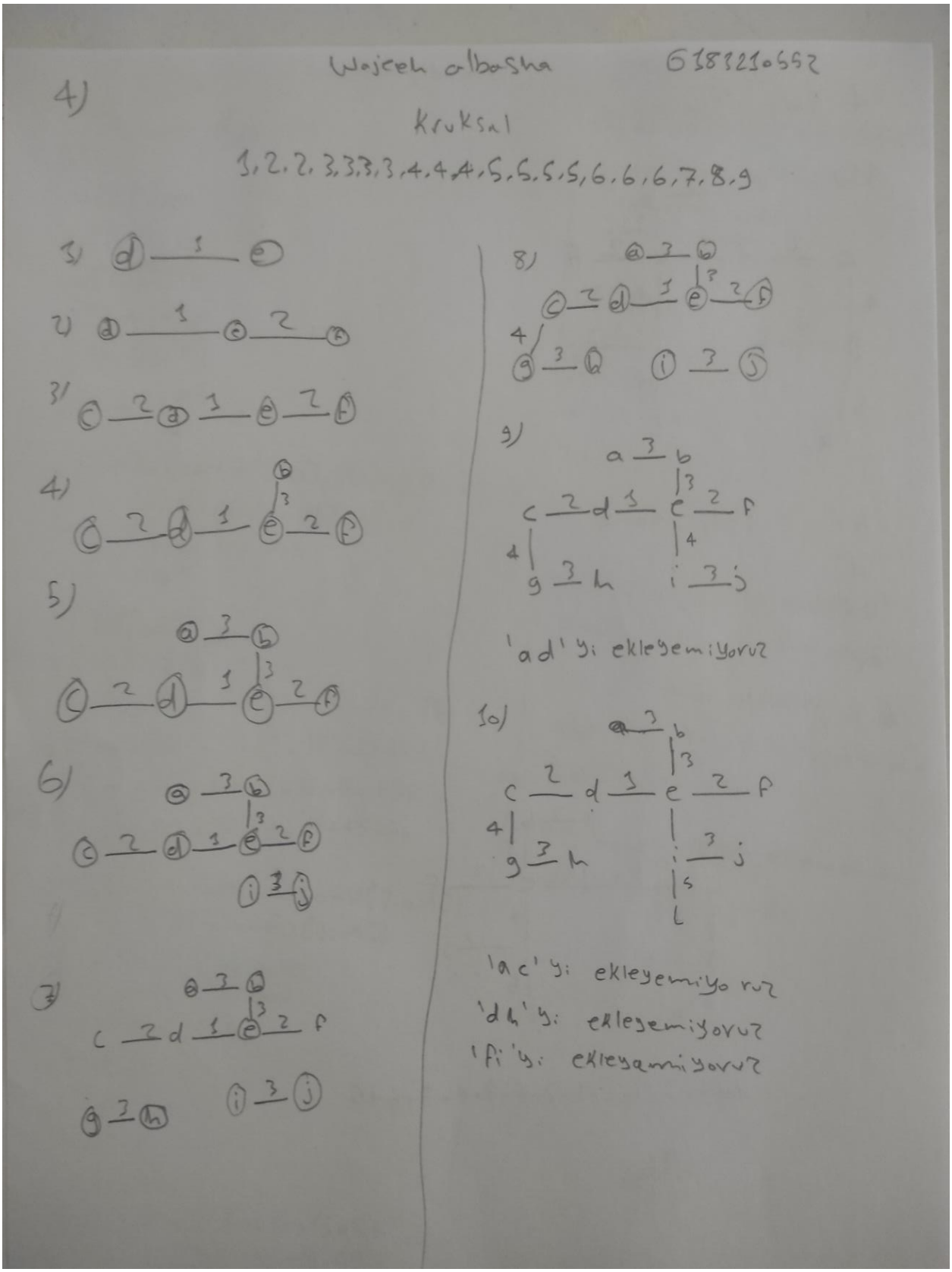
V[5,1] = 0
V[5,2] = 0
V[5,3] = 0
V[5,4] = 0
V[5,5] = 50
V[5,6] = 50

Sırt çantasına yerleştirilecek olan eşyalar 3 ve 5 olacak

3 ve 5'i yerleştirdikten sonra ağırlık 1+5 = 6, sırt çantasının kapasitesine eşit

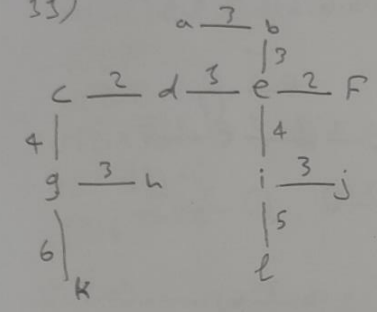
Ve bu ağırlıkları eklemenin değeri: 15 + 50 = 65

4. soru :


Wojciech albasha 618321552
4)
Kruksal
1,2,2,3,3,3,3,4,4,4,5,5,5,5,6,6,6,7,8,9
1)
2)
3)
4)
5)
6)
7)
8)
9)
'ad' yi ekleyemiyoruz
10)
'ac' yi ekleyemiyoruz
'dh' yi ekleyemiyoruz
'fi' yi ekleyemiyoruz

4. Soru devamı Kruksal

3)



'hi' yi ekleyemiyoruz

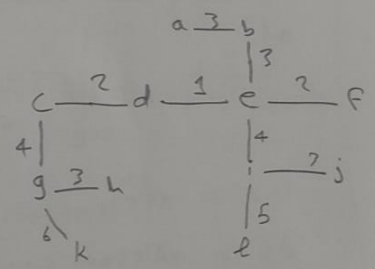
'bF' = =

'hK' = =

'KL' = =

'lj' = =

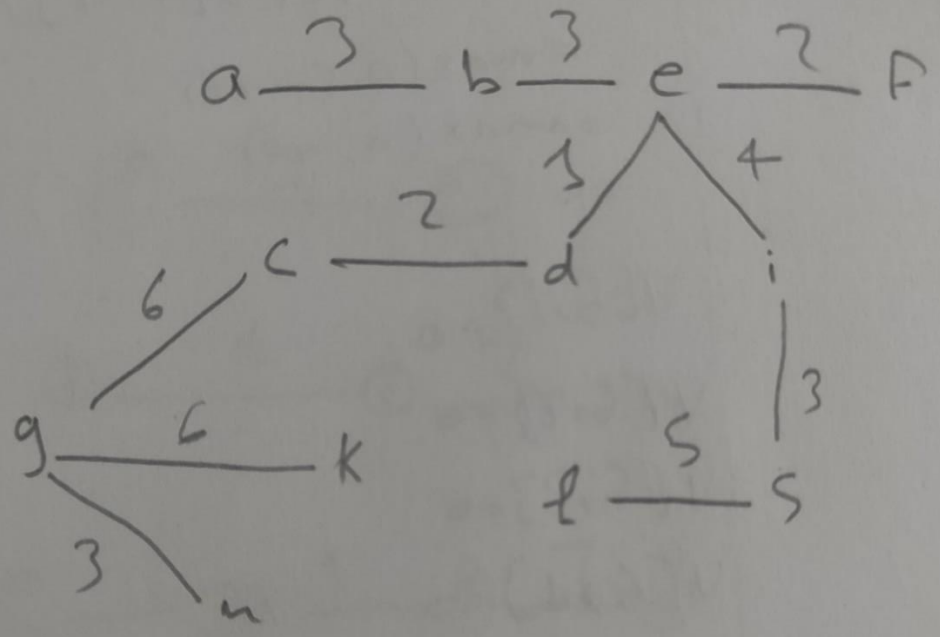
Son şekil



değer = 3+3+2+1+2+4+3+4+3+6+5

= 36

4. Soru prims

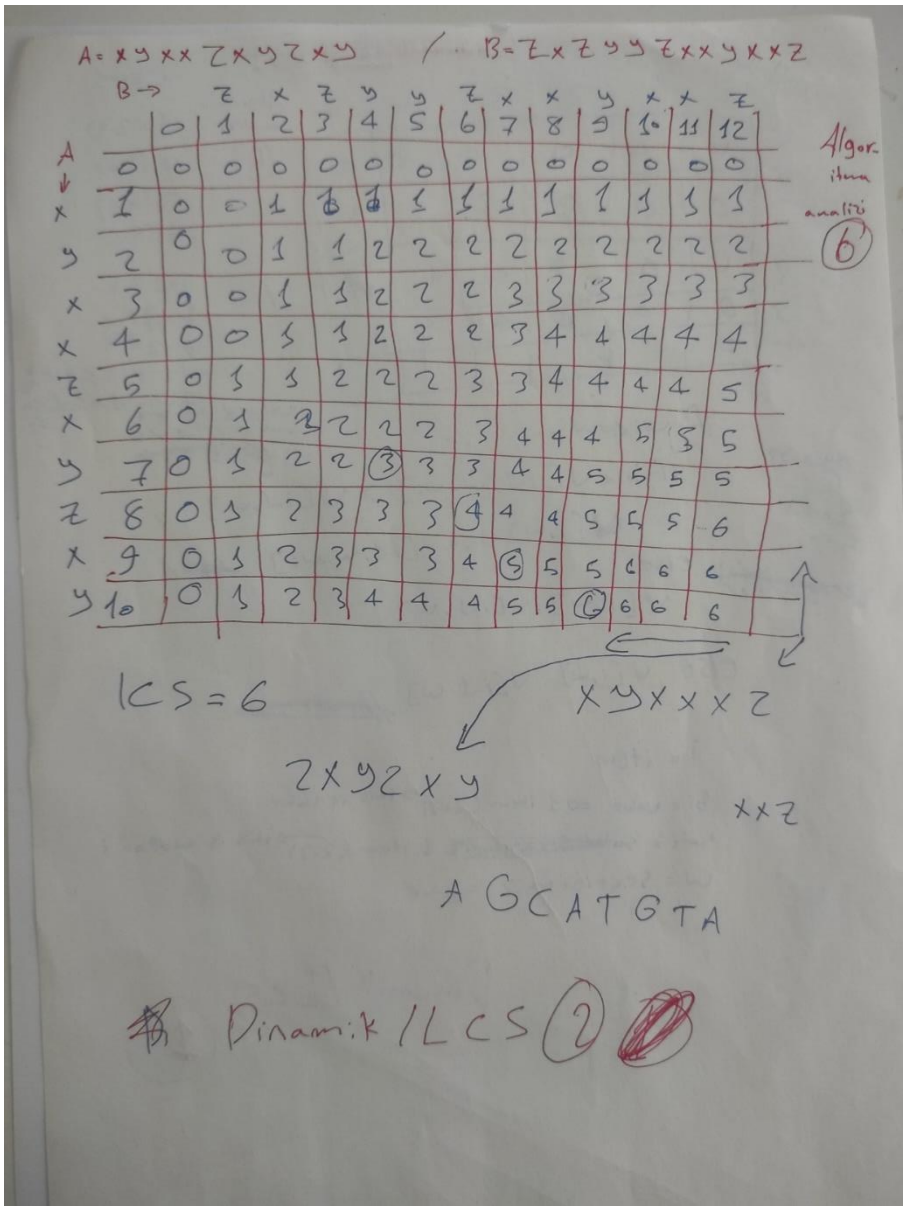


5. soru a şıkkı :

```python
def lcs(s1, s2):
    len_s1, len_s2 = len(s1) + 1, len(s2) + 1
    S = [[0] * len_s1 for _ in range(len_s2)]

    for i in range(1, len_s2):
        for j in range(1, len_s1):
            if s1[j - 1] == s2[i - 1]:
                S[i][j] = S[i - 1][j - 1] + 1
            else:
                S[i][j] = max(S[i - 1][j], S[i][j - 1])

    i, j = len_s2 - 1, len_s1 - 1
    result = ""
    while i > 0 and j > 0:
        if s1[j - 1] == s2[i - 1]:
            result += s1[j - 1]
            i, j = i - 1, j - 1
        elif S[i - 1][j] > S[i][j - 1]:
            i -= 1
        else:
            j -= 1

    return S[len_s2 - 1][len_s1 - 1], result[::-1]

if __name__ == '__main__':
    X = "xyxxzxyzxy"
    Y = "zxzyyzxxyxxz"
    print(lcs(X, Y))
```

Bu algoritmanın zaman karmaşıklığı `O(nm)`'dir.

Koddaki örnek çözümü :


A= xyxxzxyzxy
B= zxzyyzxxyxxz
B→
A↓
Algoritma analizi
6
LCS=6
xyxxxz
zxyzxy
xxz
AGCATGTA
Dinamik LCS (2)

5. soru b şıkkı :

```cpp
//5. soru b

#include <bits/stdc++.h>
using namespace std;
// Recursively print the arrangement for minimum cost of multiplication
void printBracketsMatrixChain(int i, int j, vector<vector<int>> &brackets, char &cur_name){

    // you have a single matrix ( you cannot further reduce the problem, so print the matrix )
    if(i == j){
        cout<<cur_name;
        cur_name++;
    } else {
        cout<<"(";

        // Reduce the problem into left sub-problem ( left of optimal arrangement )
        printBracketsMatrixChain(i, brackets[i][j], brackets, cur_name);

        // Reduce the problem into right sub-problem ( right of optimal arrangement )
        printBracketsMatrixChain(brackets[i][j]+1, j, brackets, cur_name);
        cout<<")";
    }
}
void matrixMultiplicationProblem(vector<int> matrixSize) {
    int numberOfMatrices = matrixSize.size()-1;
    // dp[i][j] = minimum number of operations required to multiply matrices i to j
    int dp[numberOfMatrices][numberOfMatrices];
    // initialising dp array with INT_MAX ( maximum number of operations )
    for(int i=0;i<numberOfMatrices;i++){
        for(int j=0;j<numberOfMatrices;j++){
            dp[i][j] = INT_MAX;
            if(i == j) // for a single matrix from i to i, cost = 0
                dp[i][j] = 0;
        }
    }
    vector<vector<int>> brackets(numberOfMatrices, vector<int>(numberOfMatrices, 0));
    for(int len=2;len<=numberOfMatrices;len++){
        for(int i=0;i<numberOfMatrices-len+1;i++){
            int j = i+len-1;
            for(int k=i;k<j;k++) {
                int val = dp[i][k]+dp[k+1][j]+(matrixSize[i]*matrixSize[k+1]*matrixSize[j+1]);
                if(val < dp[i][j]) {
                    dp[i][j] = val;
                    brackets[i][j] = k;
                }
            }
        }
    }
    // naming the first matrix as A
    char cur_name = 'A';
```

```
    // naming the first matrix as A
    char cur_name = 'A';

    // calling function to print brackets
    printBracketsMatrixChain(0, numberOfMatrices-1, brackets, cur_name);
    cout<<endl;
}
int main() {
    int t;cin>>t;
    while(t--) {
        int n; cin>>n;
        vector<int> matrixSize(n);
        for(int i=0;i<n;i++)cin>>matrixSize[i];
        matrixMultiplicationProblem(matrixSize);
    }
}
```

Karmaşıklık Analizi

Zaman Karmaşıklığı: O (N ^ 3)

Burada, O (N ^ 2) 'de çalışan matrisler için sınır görevi gören iki i ve j işaretçisi ele alıyoruz. Dış döngülerin içindeki iç içe geçmiş döngü doğrusal zaman O (N) alır. Yani algoritmanın çalışmasına neden oluyor **O (N ^ 3)** toplamda.