Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

JHANVI PAREKH

60009210033

CSE(Data Science)

Experiment 6

(Dynamic Programming)

Aim: Implementation of coin change problem using dynamic programming.

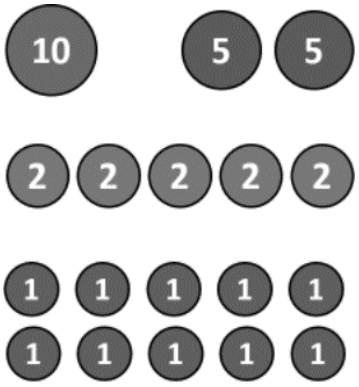
Theory:

Making Change problem is to find change for a given amount using a minimum number of coins from a set of denominations.

Explanation: If we are given a set of denominations D = {d0, d1, d2, …, dn} and if we want to change for some amount N, many combinations are possible. Suppose {d1, d2, d5, d8}, {d0, d2, d4}, {d0, d5, d7} all feasible solutions.

The aim of making a change is to find a solution with a minimum number of coins / denominations. Clearly, this is an optimization problem.

This problem can also be solved by using a greedy algorithm. However, greedy does not ensure the minimum number of denominations.



Various denominations for amount 10

General assumption is that infinite coins are available for each denomination. We can select any denomination any number of times.

Mathematical Formulation:

$$C[i,j] = \begin{cases} 1 + C[1, j - d_i], & \text{, if } i = j \\ C[i-1, j], & \text{if } j < d_i \\ \min(C[i-1, j], 1 + C[i, j - d_i]), & \text{otherwise} \end{cases}$$

Pseudocode:

Department of Computer Science and Engineering (Data Science)

```
Algorithm MAKE_A_CHANGE(d,N)
// d[1…n] = [d1,d2,…,dn] is array of n denominations


// C[1…n, 0…N] is n x N array to hold the solution of sub problems
// N is the problem size, i.e. amount for which change is required

for i ← 1 to n do
  C[i, 0] ← 0
end

for i ← 1 to n do
  for  j ← 1 to N do
    if i = = 1 & & j < d [i] then
          C[i, j] ←  ∞
    else if i == 1 then
          C[i, j] ← 1 + C[1, j – d[1])
    else if j < d [i] then
          C[i, j] ←  C[I – 1, j]
    else
          C[i, j] ← min (C[i – 1, j] ,1 + C[i, j – d[i])
    end
end
end
return C[n, N]

Algorithm TRACE_MAKE_A_CHANGE(C)
// When table C is filled up, i = n and j = N

Solution = { }
while ( j > 0 ) do
  if (C[i, j] == C[i – 1, j] then
    i ← i – 1
  else
    j ← j – di
    Solution = Solution U {d[i] }
  end
end
```

Complexity:

Best Case Time Complexity: O(n)

Lab Assignment:

Write a C Program and consider the set of denominations, D=1,4,6. Achieve the sum of 8 and calculate the number of coins required and the actual denominations needed using dynamic programming.

CODE:

```
#include <stdio.h>
```

Department of Computer Science and Engineering (Data Science)

```
#include <limits.h>
#define N 3
#define SUM 8
int min(int a, int b) {
return a < b ? a : b;
}
void findCoins(int coins[], int dp[][SUM+1]) {
int i = N, j = SUM, count = 0;
while (j > 0) {
if (dp[i][j] == dp[i-1][j]) {
i--;
} else {
coins[count++] = i;
j -= i;
}
}
}
int main() {
int denominations[N] = {1, 4, 6};
int dp[N+1][SUM+1], coins[SUM], count = 0;
for (int i = 0; i <= N; i++) {
for (int j = 0; j <= SUM; j++) {
if (j == 0) {
dp[i][j] = 0;
} else {
dp[i][j] = INT_MAX;
}
}
}
for (int i = 1; i <= N; i++) {
for (int j = 1; j <= SUM; j++) {
if (j < denominations[i-1]) {
dp[i][j] = dp[i-1][j];
} else {
dp[i][j] = min(dp[i-1][j], 1 + dp[i][j-denominations[i-1]]);
}
}
}
printf("60009210033 Jhanvi Parekh\n");
printf("Number of coins required: %d\n", dp[N][SUM]);
findCoins(coins, dp);
printf("Actual denominations needed: ");
for (int i = 0; i < dp[N][SUM]; i++) {
printf("%d ", denominations[coins[i]-1]);
}
return 0;
}
```

main.c

```c
#include <stdio.h>
#include <limits.h>
#define N 3
#define SUM 8
int min(int a, int b) {
return a < b ? a : b;
}
void findCoins(int coins[], int dp[][SUM+1]) {
int i = N, j = SUM, count = 0;
while (j > 0) {
if (dp[i][j] == dp[i-1][j]) {
i--;
} else {
coins[count++] = i;
j -= i;
}
}
}
int main() {
int denominations[N] = {1, 4, 6};
int dp[N+1][SUM+1], coins[SUM], count = 0;
for (int i = 0; i <= N; i++) {
for (int j = 0; j <= SUM; j++) {
if (j == 0) {
dp[i][j] = 0;
} else {
dp[i][j] = INT_MAX;
}
}
}
for (int i = 1; i <= N; i++) {
for (int j = 1; j <= SUM; j++) {
if (j < denominations[i-1]) {
dp[i][j] = dp[i-1][j];
```

Department of Computer Science and Engineering (Data Science)

main.c

```
j -= i;
}
}
}
int main() {
int denominations[N] = {1, 4, 6};
int dp[N+1][SUM+1], coins[SUM], count = 0;
for (int i = 0; i <= N; i++) {
for (int j = 0; j <= SUM; j++) {
if (j == 0) {
dp[i][j] = 0;
} else {
dp[i][j] = INT_MAX;
}
}
}
for (int i = 1; i <= N; i++) {
for (int j = 1; j <= SUM; j++) {
if (j < denominations[i-1]) {
dp[i][j] = dp[i-1][j];
} else {
dp[i][j] = min(dp[i-1][j], 1 + dp[i][j-denominations[i-1]]);
}
}
}
printf("60009210033 Jhanvi Parekh\n");
printf("Number of coins required: %d\n", dp[N][SUM]);
findCoins(coins, dp);
printf("Actual denominations needed: ");
for (int i = 0; i < dp[N][SUM]; i++) {
printf("%d ", denominations[coins[i]-1]);
}
return 0;
}
```

Output

```
/tmp/gJE9jFWs5r.o
60009210033 Jhanvi Parekh
Number of coins required: 2
Actual denominations needed: 4 4
```

* Coin change Problem using DP

$$c[i,j] = \begin{cases} 1 + c[i, j-di] & \text{if } i == 1,\ i = j \\ c[i-1, j] & \text{if } j < di \\ \min(c[i-1, j],\ 1 + c[i, j-di]) & \text{otherwise} \end{cases}$$

where di is the i^{th} denomination,
j is the size of the sub problem,
c[n, N] = solution of the problem
where n = number of denominations
N = amount for which the change is required

Q1. Consider the set of denominations where D = 1, 4, 6, achieve the sum of '8' and calculate the number of coins required and the actual denominations using DP.

i di | j	0	1	2	3	4	5	6	7	8
i = 0	0	0	0	0	0	0	0	0	0
i = 1, di = 1	0	1	2	3	4	5	6	7	8
i = 2, di = 4	0	1	2	3	1	2	3	4	2
i = 3, di = 6	0	1	2	3	1	2	1	2	2

c[1, 1] = i = 1, di = 1, j = 1
case 1 accepted ∴ i = j, i == 1
c[1,1] = 1 + c[1, j - di]
= 1 + [1, 1 - 1]
= 1 + [1, 0]
= 1 + c[j, i]
= 1 + 0
= 1

Department of Computer Science and Engineering (Data Science)

PAGE :
DATE : / /

c[2,1] = i = 2 , d2 = 4, j = 1

case 2 accepted

c[2,1] = c[2-i, 1]

= c[1,1] = 1

c[3,1] = i = 3, j = 1, d1 = 6

case 2 : c[i-1, j]

c[3-1, 1]

c[2,1]

1

c[1,2] = i = 1, d1 = 1, j = 2

case 1 : 1 + c[1, 2-1]

= 1 + c[1,1]

1 + 1

2

c[2,2] : i = 2, j = 2, di = 4

case 2 : c[i-1, j]

c[2-1, 2]

c[1,2]

2

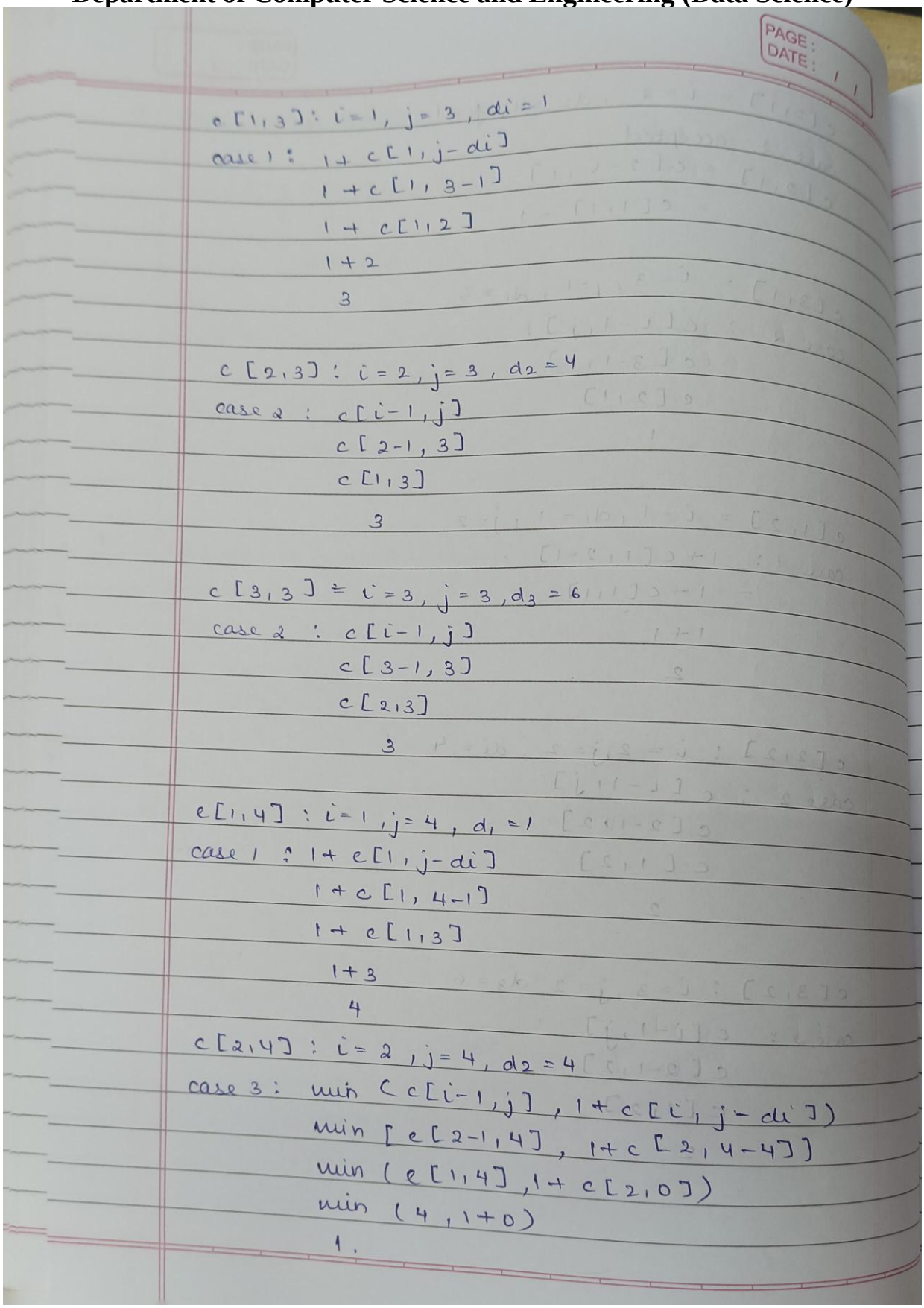
c[3,2] : i = 3, j = 2, d3 = 6

case 2 : c[i-1, j]

c[2-1, 2]

c[1,2]

2


PAGE :
DATE : / /
c[1,3] : i=1, j=3, di=1
case 1 : 1+c[1,j-di]
1+c[1,3-1]
1+c[1,2]
1+2
3
c[2,3] : i=2, j=3, d2=4
case 2 : c[i-1,j]
c[2-1,3]
c[1,3]
3
c[3,3] : i=3, j=3, d3=6
case 2 : c[i-1,j]
c[3-1,3]
c[2,3]
3
c[1,4] : i=1, j=4, d1=1
case 1 : 1+c[1,j-di]
1+c[1,4-1]
1+c[1,3]
1+3
4
c[2,4] : i=2, j=4, d2=4
case 3 : min (c[i-1,j], 1+c[i,j-di])
min [c[2-1,4], 1+c[2,4-4]]
min (c[1,4], 1+c[2,0])
min (4, 1+0)
1.

PAGE :
DATE : / /

c [3, 4] : i = 3, j = 4, d_3 = 6
case 2 : c [i-1, j]
c [2, 4]
1

c [1, 5] : i = 1, j = 5, d_1 = 1
case 1 : 1 + c [1, j - di]
1 + c [1, 5-1]
1 + c [1, 4]
1 + 4
5

c [2, 5] : i = 2, j = 5, d_2 = 4
case 3 : min (c [2-1, 5], 1 + c [2, 1])
min (c [1, 5], 1 + c [2, 1])
min (5, 1+1)
2

c [3, 5] : i = 3, j = 5, d_3 = 6
case 2 : c [i-1, j]
c [3-1, 5]
c [2, 5]
2

c [1, 6] : i = 1, j = 6, d_1 = 1
case 1 : 1 + c [1, j - di]
1 + c [1, 5]
1 + 5
6

c[2,6] : i=2, j=6, d_2 = 4
case 3 : min (c[i-1, j], 1+c[i, j-di])
min (c[1,6], 1+c[2,2]) = min (6, 1+
3

c[3,6] : i=3, j=6, d_1 = 6
case 3 : min (c[i-1, j], 1+c[i, j-di])
min (c[2,6], 1+c[3,0])
min (3, 0)
0

c[1,7] : i=1, j=7, d_1 = 1
case 1 : 1+c[1, j-di]
1+c[1,6]
1+6
7

c[2,7] : i=2, j=7, d_2 = 4
case 3 : min (c[i-1, j], 1+c[i, j-di])
min (c[1,7], 1+c[2,3])
min (7, 1+3)
4

c[3,7]
c[1,8] : i=3, j=8, d_2 = 6
case 3 : min (c[i-1, j], 1+c[i, j-di])
min (c[2,7], 1+c[3,1])
min (4, 1+1)
2.

PAGE :
DATE : / /

c[1,8] : i=1 , j=8 , d_1=1

case 1 : 1 + c[1, j-di]

1 + c[1, 8-1]

1 + c[1,7]

1 + 7

8

c[2,8] : i= 2, j=8 , d_2 = 4

case 3 : min (c[i-1, j] , 1 + c[i, j-di])

min (c[1,8] , 1+c[2,4])

min (8, 1+1)

2

c[3,8] : i=3, j=8 , d_3 = 6

case 2: c[i-1, j]

c[2,8]

2

From the last cell of table

i = 3 , j = 8

check c[i,j] = c[i-1,j]

as nothing is added in the previous solution go to the previous step by reducing the value of i by 1

i = i-1

= 3-1

i = 2

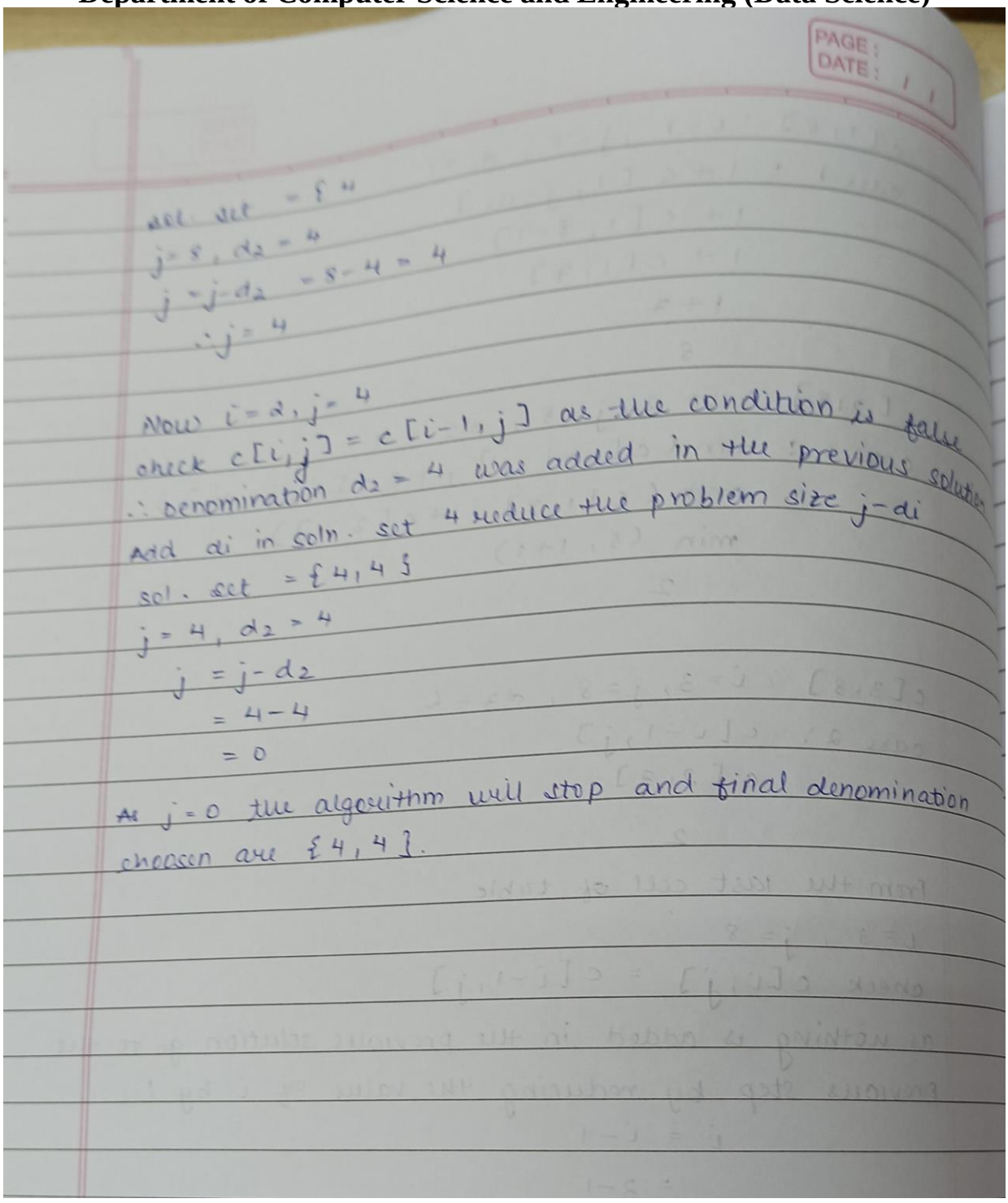
Now , i = 2 , j = 8

check c[i,j] = c[i-1,j]

as the condition is false

∴ Denomination di = d_2 = 4 was added in the previous solution

Add di in solution set 4 reduce the problem size j by di ↳ (j-di)


PAGE :
DATE : / /
sol. set = { 4
j = 8, d2 = 4
j = j - d2 = 8 - 4 = 4
∴ j = 4
Now i = 2, j = 4
check c[i,j] = c[i-1, j] as the condition is false
∴ denomination d2 = 4 was added in the previous solution
Add di in soln. set 4 reduce the problem size j-di
sol. set = { 4, 4 }
j = 4, d2 = 4
j = j - d2
= 4 - 4
= 0
At j = 0 the algorithm will stop and final denomination
choosen are { 4, 4 }.