

Design and Analysis of Algorithms Lab

Assessment-2

Name: Armaan Indani

Reg. No.: 22BCE3347

Slot : L41+42

Faculty: Dr. Joshva Devdas T – 16700

Question 1:

SCOPE / BCSE204P_VL2023240504902 / ASSESSMENT LAB 2



ASSESSMENT LAB 2

Back

Question 1

Correct

Marked out of
2.50

Flag question

Write a program to apply LC Branch and Bound to find out the maximum profit gain by the thief using the following information for the 0/1 knap sack problem.

$N=4$, $M=15$ Profit = (10,10,12,18) weight = (2, 4, 6, 9)

For example:

Test	Input	Result
1	4 15 10 10 12 18 2 4 6 9	Profit =38 solution = {1,1,0,1}

Answer: (penalty regime: 0 %)

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <limits.h>
4 #include <stdlib.h>
5
6 float UB = 0;
7
8 struct node
9 {
10     float cost;
11     char items[4];
12     int item_no;
13 };
14
15 struct node *newNode(char it[4], int it_no)
16 {
17     struct node *node1 = (struct node *)malloc(sizeof(struct node));
18     node1->cost = INT_MIN;
19     strncpy(node1->items, it, sizeof(node1->items));
20     node1->item_no = it_no;
21     return node1;
22 }
23
24 struct node *ans;
25
26 struct Queue
27 {
28     struct node *n[100];
29     int front;
30     int back;
31 };
32
33 void enqueue(struct Queue *Q, struct node *node1)
34 {
35     Q->front += 1;
36     Q->n[Q->front] = node1;
37 }
38
```

```

39 void dequeue(struct Queue *Q)
40 {
41     Q->back += 1;
42 }
43
44 float ub_cost_calc(int n,int m,int p[],int w[],struct node *node1)
45 {
46     char s[n];
47     strncpy(s, node1->items, sizeof(s));
48     int wt = 0;
49     float pf = 0;
50     int i;
51     for (i = 0; i < n; i++)
52     {
53         if (s[i] == '1')
54         {
55             if (wt + w[i] <= m)
56             {
57                 wt += w[i];
58                 pf += p[i];
59             }
60             else
61             {
62                 break;
63             }
64         }
65     }
66
67     if (UB > pf)
68     {
69         UB = pf;
70     }
71     if (wt < m && i < n)
72     {
73         float extp = (float)p[i] / w[i];
74         pf -= (m - wt) * extp;
75     }
76     return pf;
77 }
78

```

```

79 void addToQueue(int n,int m,struct node *root,struct Queue *Q,int p[n]
80 {
81     if (!root)
82     {
83         return;
84     }
85     int it_no = root->item_no;
86     if (it_no >= n)
87     {
88         return;
89     }
90     char it[4];
91     strncpy(it, root->items, sizeof(it));
92
93     it[it_no] = '0';
94
95     // Check if cost > UB
96
97     root->cost = ub_cost_calc(n, m, p, w, root);
98
99     if (root->cost > UB)
100     {
101         return;
102     }
103
104     ans = root;
105
106     for (int i = it_no + 1; i < n; i++)
107     {
108         char it1[4];
109         strncpy(it1, root->items, sizeof(it1));
110         it1[i] = '0';
111         struct node *newN = newNode(it1, i);
112         enqueue(Q, newN);
113     }
114 }
115
116 int main()
117 {
118     int n, m;
119     scanf("%d", &n);

```

```

120     scanf("%d", &m);
121     int p[n], w[n];
122
123     for (int i = 0; i < n; i++)
124     {
125         scanf("%d", &p[i]);
126     }
127     for (int i = 0; i < n; i++)
128     {
129         scanf("%d", &w[i]);
130     }
131
132     struct node *root = newNode("1111", 0);
133
134     struct Queue *Q = (struct Queue *)malloc(sizeof(struct Queue));
135
136     Q->front = -1;
137     Q->back = 0;
138
139     enqueue(Q, root);
140
141     while (Q->front >= Q->back)
142     {
143         addToQueue(n, m, Q->n[Q->back], Q, p, w);
144         dequeue(Q);
145     }
146     int total_profit = 0;
147     for (int i = 0; i < n; i++)
148     {
149         if (ans->items[i] == '1')
150         {
151             total_profit += p[i];
152         }
153     }
154     printf("Profit = %d\n", total_profit);
155     printf("solution = {%c", ans->items[0]);
156     for (int i = 1; i < n; i++)
157     {
158         printf(", %c", ans->items[i]);
159     }
160     printf(")");

```

```

161 }

```

	Test	Input	Expected	Got	
✓	1	4 15 10 10 12 18 2 4 6 9	Profit =38 solution = {1,1,0,1}	Profit =38 solution = {1,1,0,1}	✓

Passed all tests! ✓

Question 2

SCOPE / BCSE204P_VL2023240504902 / ASSESSMENT LAB 2



ASSESSMENT LAB 2

Back

Question 2

Correct

Marked out of
2.50

🚩 Flag question

Apply Branch and Bound technique to solve the Job Selection Problem by writing a program using the following information

Number of Jobs = 4

Jobs	1	2	3	4
Penalty	5	10	6	3
Deadline	1	3	2	1
Time	1	2	1	1

For example:

Test	Result
1	Cost = 5 Upper Bound = 8 Jobs Completed within deadline are J2 and J3

Answer: (penalty regime: 0 %)

```
1  #include <stdio.h>
2  #include <limits.h>
3  #include <string.h>
4  #include <stdlib.h>
5
6  int UB = INT_MAX;
7
8  struct node
9  {
10     int cost;
11     char jobs[4];
12     int job_no;
13 };
14
15 struct node *ans;
16
17 struct Queue
18 {
19     int front;
20     int back;
21     struct node *n[100];
22 };
23
24 void enqueue(struct Queue *Q, struct node *n)
25 {
26     Q->front += 1;
27     Q->n[Q->front] = n;
28 }
29
30 void dequeue(struct Queue *Q)
31 {
32     Q->back += 1;
33 }
34
35 struct node *newNode(char j[4], int jn)
36 {
37     struct node *node1 = (struct node *)malloc(sizeof(struct node));
38     node1->cost = 0;
39     strncpy(node1->jobs, j, sizeof(node1->jobs));
40     node1->job_no = jn;
```

```

41     return node1;
42 }
43
44 void addToQueue(int n, struct node *root, struct Queue *Q, int penalty)
45 {
46     if (!root)
47     {
48         return;
49     }
50     int jn = root->job_no;
51     if (jn >= n)
52     {
53         return;
54     }
55     char j[4];
56     int local_ub = 0;
57     strncpy(j, root->jobs, sizeof(j));
58
59     j[jn] = '1';
60
61     int total_time = 0;
62     for (int i = 0; i < n; i++)
63     {
64         if (j[i] == '1')
65         {
66             total_time += time[i];
67         }
68     }
69     int possible = 0;
70     for (int i = 0; i < n; i++)
71     {
72         if (j[i] == '1')
73         {
74             if (total_time <= deadline[i])
75             {
76                 possible = 1;
77                 break;
78             }
79         }
80     }

```

```

81     if (!possible)
82     {
83         return;
84     }
85
86     for (int i = 0; i < jn; i++)
87     {
88         if (j[i] == '0')
89         {
90             root->cost += penalty[i];
91             local_ub += penalty[i];
92         }
93     }
94     for (int i = jn + 1; i < n; i++)
95     {
96         local_ub += penalty[i];
97     }
98
99     UB = local_ub > UB ? UB : local_ub;
100
101     if (root->cost > UB)
102     {
103         return;
104     }
105
106     ans = root;
107
108     for (int i = jn + 1; i < n; i++)
109     {
110         char j1[4];
111         strncpy(j1, root->jobs, sizeof(j1));
112         j1[i] = '1';
113         struct node *newN = newNode(j1, i);
114         enqueue(Q, newN);
115     }
116 }
117
118 int main()
119 {
120     int n = 4;

```



```

121     int penalty[4] = {5, 10, 6, 3};
122     int deadline[4] = {1, 3, 2, 1};
123     int time[4] = {1, 2, 1, 1};
124
125     struct Queue *Q = (struct Queue *)malloc(sizeof(struct Queue));
126     Q->front = -1;
127     Q->back = 0;
128     struct node *root = newNode("0000", 0);
129     enqueue(Q, root);
130
131     while (Q->front >= Q->back)
132     {
133         addToQueue(n, Q->n[Q->back], Q, penalty, deadline, time);
134         Q->back++;
135     }
136
137     printf("Cost = %d\n", ans->cost);
138     printf("Upper Bound = %d\n", UB);
139     printf("Jobs Completed within deadline are ");
140
141     int first = 1;
142     for (int i = 0; i < n; i++)
143     {
144         if (ans->jobs[i] == '1')
145         {
146             if (first)
147             {
148                 first = 0;
149                 printf("J%d ", i + 1);
150             }
151             else
152             {
153                 printf("and J%d ", i + 1);
154             }
155         }
156     }
157     return 0;
158 }

```

	Test	Expected	Got
✓	1	Cost = 5 Upper Bound = 8 Jobs Completed within deadline are J2 and J3	Cost = 5 Upper Bound = 8 Jobs Completed within dead

Passed all tests! ✓

Question 3

SCOPE / BCSE204P_VL2023240504902 / ASSESSMENT LAB 2



ASSESSMENT LAB 2

Back

Question 3

Correct

Marked out of
2.50

Flag question

Write a program to apply KMP String Matching algorithm to verify the given pattern is present in the string or not. If present display its occurrences

String b a c b a b a b a b a c a a b
Pattern a b a b a c a

For example:

Test	Input	Result
1	b a c b a b a b a b a c a a b a b a b a c a	Pattern is found in the string Number of Shifts needed is 6

Answer: (penalty regime: 0 %)

```
1 #include <stdio.h>
2 #include <string.h>
3
4 int max(int a, int b)
5 {
6     return a > b ? a : b;
7 }
8
9 int main()
10 {
11     char str1[40];
12     char pat1[20];
13     char str[20];
14     char pat[10];
15
16     fgets(str1, 40, stdin);
17     fgets(pat1, 20, stdin);
18
19     for (int i = 0; i < strlen(str1) / 2; i++)
20     {
21         str[i] = str1[2 * i];
22     }
23     for (int i = 0; i < strlen(pat1) / 2; i++)
24     {
25         pat[i] = pat1[2 * i];
26     }
27     pat[strlen(pat) - 1] = '\0';
28
29     // printf("%ld %ld\n", strlen(pat), strlen(str));
30
31     int pi_table[strlen(pat)];
32     // puts(str);
33     // printf("*****\n");
34     // puts(pat);
35
36     for (int i = 0; i < strlen(pat); i++)
37     {
38         pi_table[i] = 0;
39     }
40
```

```

41     for (int i = 1; i < strlen(pat); i++)
42     {
43         if (pat[0] == pat[i])
44         {
45             for (int j = 0; j < strlen(pat) - i; j++)
46             {
47                 if (pat[j] == pat[i + j])
48                 {
49                     pi_table[i + j] = max(pi_table[i + j], j + 1);
50                 }
51                 else
52                 {
53                     break;
54                 }
55             }
56         }
57     }
58
59     // for(int i=0; i<strlen(pat); i++)
60     // {
61     //     printf("%d ", pi_table[i]);
62     // }
63
64     int i = 0, j = 0;
65     while (i < strlen(str))
66     {
67         // printf("[%d %d] ",i,j);
68         if (j == strlen(pat))
69         {
70             printf("Pattern is found in the string\n");
71             printf("Number of Shifts needed is %ld", i - strlen(pat));
72             return 0;
73         }
74         if (str[i] == pat[j])
75         {
76             i++;
77             j++;
78         }
79         else if (j == 0)
80         {
81             i++;
82         }
83         else
84         {
85             j = pi_table[j - 1];
86         }
87     }
88     printf("Pattern is not found in the string");
89     return 0;
90 }

```

	Test	Input	Expected	Got
✓	1	b a c b a b a b a b a c a a b a b a b a c a	Pattern is found in the string Number of Shifts needed is 6	Pattern i Number of

◀

▶

Passed all tests! ✓

Question 4

SCOPE / BCSE204P_VL2023240504902 / ASSESSMENT LAB 2



ASSESSMENT LAB 2

Back

Question 4

Correct

Marked out of
2.50

🚩 Flag question

Write a program to apply the Rabin Karp String Matching algorithm to check whether the given pattern is present in the String or not

Note : Use the Robin Karp Finger Print function to verify the pattern is present in the string or not

For example:

Test	Input	Result
1	c c a c c a a e d b a d b a	The given pattern is present in the String

Answer: (penalty regime: 0 %)

```
1 #include <stdio.h>
2 #include <string.h>
3
4 int hash_fn(char *start, int len)
5 {
6     int hash_val = 0;
7     for (int i = 0; i < len; i++)
8     {
9         hash_val = 10 * hash_val + ((int)start[i] - 96);
10    }
11    return hash_val;
12 }
13
14 int check_str(char *pat, char *str, int len)
15 {
16     for (int i = 0; i < len; i++)
17     {
18         if (pat[i] != str[i])
19         {
20             return 0;
21         }
22     }
23     return 1;
24 }
25
26 int main()
27 {
28     char str1[25];
29     char pat1[10];
30     char str[15];
31     char pat[5];
32
33     fgets(str1, 25, stdin);
34     fgets(pat1, 10, stdin);
35
36     for (int i = 0; i < strlen(str1) / 2; i++)
37     {
38         str[i] = str1[2 * i];
39     }
40     for (int i = 0; i < strlen(pat1) / 2; i++)
```

```

41 {
42     pat[i] = pat1[2 * i];
43 }
44
45 int pat_len = strlen(pat);
46 int hash_p = hash_fn(&pat[0], strlen(pat));
47
48 for (int i = 0; i < strlen(str) - pat_len + 1; i++)
49 {
50     int hash_str = hash_fn(&str[i], strlen(pat));
51     if (hash_str == hash_p)
52     {
53         if (check_str(&pat[0], &str[i], pat_len))
54         {
55             printf("The given pattern is present in the String");
56             return 0;
57         }
58     }
59 }
60 return 0;
61 }

```

	Test	Input	Expected	Got
✓	1	c c a c c a a e d b a d b a	The given pattern is present in the String	The g

Passed all tests! ✓