DAA Lab-4

01/02/25 – 230962326– Davasam Karthikeya – AIMLB

Solved question (Knapsack):

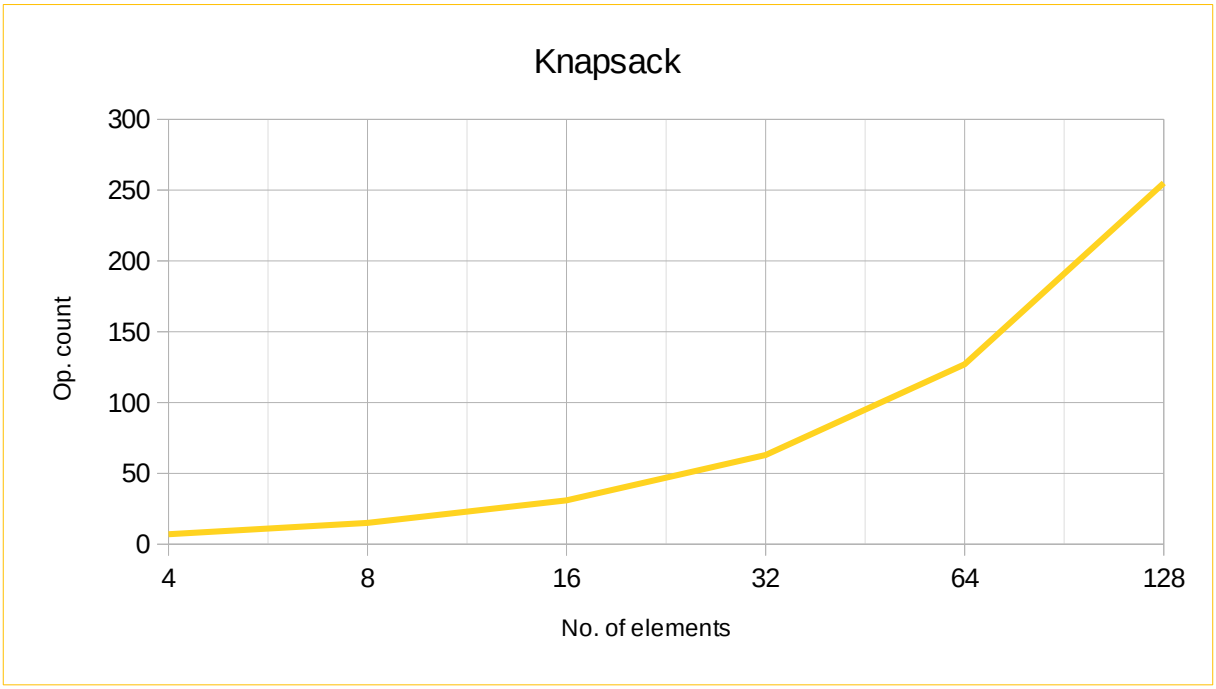
```
#include <stdio.h>
#include <stdlib.h>

int Knapsack(unsigned int *w, unsigned int *v, unsigned int n, unsigned int B) {
    unsigned int totalWeight, totalValue, index;
    unsigned int maxVal = 0, maxSequence = 0, limit = 1;
    int opcount = 0;

    for(int i = 0; i < n; i++)
        limit *= 2;

    for(unsigned int i = 1; i < limit; i++) {
        ++opcount;

        unsigned int temp = i;
        totalWeight = totalValue = 0;
        index = 0;

        while(temp) {
            if(temp & 0x1) {
                totalWeight = totalWeight + w[index];
                totalValue = totalValue + v[index];
            }

            index++;
            temp = temp >> 1;
        }

        if(totalWeight <= B && totalValue > maxVal) {
            maxVal = totalValue;
            maxSequence = i;
        }
    }

    printf("\nOp. count: %d\n",opcount);
    return maxSequence;
}

void main() {
    unsigned int *v,*w, i,n,knaps, B;

    printf("Enter the number of elements: ");
    scanf("%d", &n);
    v = (unsigned int *) calloc(n, sizeof(unsigned int));
    w = (unsigned int *) calloc(n, sizeof(unsigned int));

    printf("Please enter the weights: ");
    for(i=0;i<n;i++)
        scanf("%d",&w[i]);

    printf("Please enter the values: ");
    for(i=0;i<n;i++)
        scanf("%d",&v[i]);

    printf("Please enter the Knapsack capacity: ");
```

```
    scanf("%d", &B);

    knaps = Knapsack(w,v,n,B);

    printf("Knapsack contains the following items\n");

    i=0;
    while(knaps) {
        if(knaps & 0x1)
            printf("item %u value %u\n", (i+1), v[i]);
        i++;
        knaps = knaps >> 1;

    }
}
```

Output

Enter the number of elements: 3
Please enter the weights: 1 2 4
Please enter the values: 2 4 8
Please enter the Knapsack capacity: 5

Op. count: 7
Knapsack contains the following items
item 1 value 2
item 3 value 8



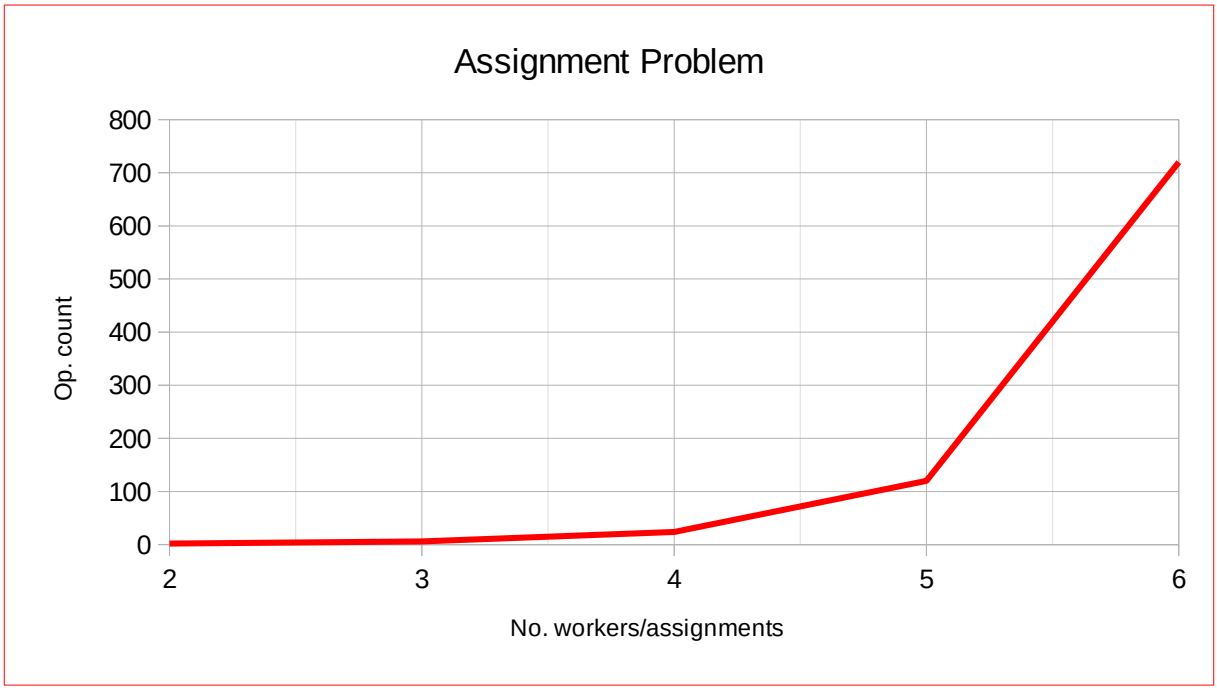
Question 1 (Assignment Problem):

```
#include <stdio.h>

#define N 4

void bfAssignment(int cost[N][N], int assign[], int start, int *minCost, int
curAssign[], int *opCount) {
    if(start == N) {
        (*opCount)++;
        int currCost = 0;
        for(int i = 0; i < N; i++)
            currCost += cost[i][assign[i]];
```

```
        if(currCost < *minCost) {
            *minCost = currCost;

            for(int i = 0; i < N; i++)
                assign[i] = curAssign[i];
        }
        return;
    }

    for (int i = start; i < N; i++) {
        int temp = curAssign[start];
        curAssign[start] = curAssign[i];
        curAssign[i] = temp;

        bfAssignment(cost, assign, start + 1, minCost, curAssign, opCount);

        temp = curAssign[start];
        curAssign[start] = curAssign[i];
        curAssign[i] = temp;
    }
}

void main() {
    int cost[N][N] = {
        {4, 2, 7, 3},
        {8, 5, 6, 9},
        {3, 7, 4, 2},
        {6, 2, 5, 8}
    };
    int assign[N];
    int curAssign[N] = {0, 1, 2, 3};
    int minCost = 99999999, opCount = 0;

    bfAssignment(cost, assign, 0, &minCost, curAssign, &opCount);

    printf("Assignment:\n");
    for (int i = 0; i < N; i++)
        printf("Worker %d -> Job %d\n", i, assign[i]);

    printf("Minimum Total Cost: %d\n", minCost);
}
```

Output

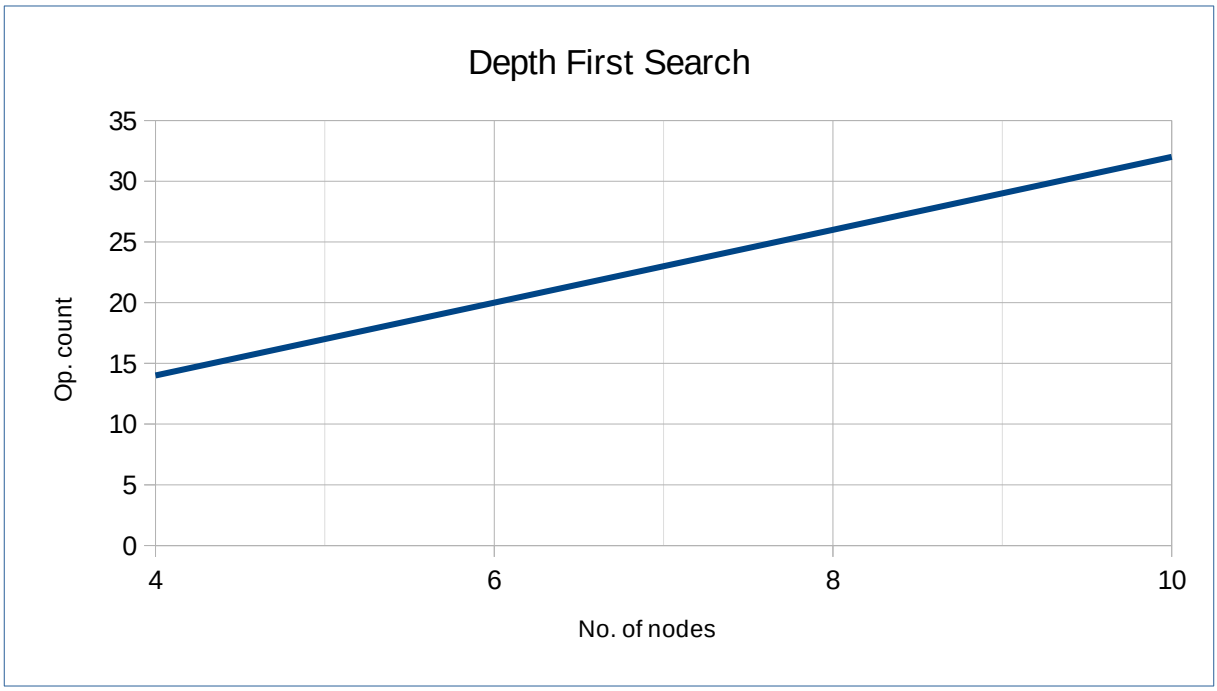
Op. count: 24
Assignment:
Worker 0 -> Job 0
Worker 1 -> Job 2
Worker 2 -> Job 3
Worker 3 -> Job 1
Minimum Total Cost: 14



Question 2 (DFS):

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

#define N 6

void dfs(int graph[N][N], int size, int root) {
    int stack[64], top = 0, j = 0;
    int *visited = calloc(size, sizeof(int)), *output = calloc(size,
sizeof(int));
    stack[top++] = root;

    while(1) {
        int len = 0;
        for(int i = 0; i < size; i++)
            len += visited[i];
        if(len >= size && top)
            break;

        int vert = stack[--top];
        printf("Pop: %d\n", vert);

        if(!visited[vert]) {
            output[j++] = vert;

            for(int i = 0; i < size; i++)
                if(graph[vert][i]) {
                    stack[top++] = i;
                    printf("Push: %d\n", i);
                }

            visited[vert] = 1;
        }
    }

    printf("Visit order: ");
    for(int i = 0; i < j; i++)
        printf("%d ", output[i]);
    printf("\n");
```

```
}

void main() {
    int graph[N][N] = {
        {0, 1, 0, 0, 1, 0},
        {1, 0, 0, 1, 0, 0},
        {0, 0, 0, 0, 1, 1},
        {0, 1, 0, 0, 0, 1},
        {1, 0, 1, 0, 0, 0},
        {0, 0, 1, 1, 0, 0}
    };

    dfs(graph, N, 2);
}
```



Output

```
Pop: 2
Push: 4
Push: 5
Pop: 5
Push: 2
Push: 3
Pop: 3
Push: 1
Push: 5
Pop: 5
Pop: 1
Push: 0
Push: 3
Pop: 3
Pop: 0
Push: 1
Push: 4
Pop: 4
Push: 0
Push: 2
Visit order: 2 5 3 1 0 4
```

Question 2 (Middle-school):

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

```
#include <stdlib.h>

#define N 6

void dfs(int graph[N][N], int size, int root) {
    int que[64], front = 0, rear = 0, j = 0;
    int *visited = calloc(size, sizeof(int)), *output = calloc(size,
sizeof(int));
    que[front++ % 64] = root;

    while(1) {
        int len = 0;
        for(int i = 0; i < size; i++)
            len += visited[i];
        if(len >= size && rear != front)
            break;

        int vert = que[rear++ % 64];
        printf("Delete: %d\n", vert);

        if(!visited[vert]) {
            output[j++] = vert;

            for(int i = 0; i < size; i++)
                if(graph[vert][i]) {
                    que[front++ % 64] = i;
                    printf("Insert: %d\n", i);
                }

            visited[vert] = 1;
        }
    }

    printf("Visit order: ");
    for(int i = 0; i < j; i++)
        printf("%d ", output[i]);
    printf("\n");
}

void main() {
    int graph[N][N] = {
        {0, 1, 0, 0, 1, 0},
        {1, 0, 0, 1, 0, 0},
        {0, 0, 0, 0, 1, 1},
        {0, 1, 0, 0, 0, 1},
        {1, 0, 1, 0, 0, 0},
        {0, 0, 1, 1, 0, 0}
    };

    dfs(graph, N, 2);
}
```

Output

Delete: 2
Insert: 4
Insert: 5
Delete: 4
Insert: 0
Insert: 2
Delete: 5
Insert: 2
Insert: 3
Delete: 0
Insert: 1

```
Insert: 4
Delete: 2
Delete: 2
Delete: 3
Insert: 1
Insert: 5
Delete: 1
Insert: 0
Insert: 3
Visit order: 2 4 5 0 3 1
```