Lab 9

4/7/24

Knapsack Problem.

```
#include <stdio.h>

int max(int a, int b){
    return (a>b)?a:b;
}

void knapsack (int W, int wt[], int val[], int n){
    int i, w;
    int K[n+1] [W+1];

    for (i=0; i<=n; i++){
        for (w=0; w<=W; w++){
            if (i==0 || w==0){
                K[i][w]=0;
            } else if (wt[i-1]<=w){
                K[i][w] = max (val[i-1]+K[i-1]
                        [w-wt[i-1]], K[i-1][w]);
            } else{
                K[i][w] =K[i-1][w];
        }
    }}
    printf("DP Table:\n");
    for (i= 0; i<=n; i++){
        for (w=0; w<=W; w++){
            printf("%4d", K[i][w]);
        }
        printf("\n");
    }
    int res = K[n][W];
    printf("\n Maximum value in knapsack=%d\n",
            res);
    printf("Items included in the knapscack:\n");
```

4/7/24

```
w=W;
for (i=n; i>0 && res >0; i--){
   if (res ==k[i-1][w])
      continue;
   else{
      printf("Item %d (Value: %d, Weight:
              %d)\n", i, val[i-1], wt[i-1]);
      res -= val[i-1];
      w -= wt[i-1];
   }
}}

int main(){
   int n=4;
   int val[] = {12, 10, 20, 15};
   int wt[] = {2, 1, 3, 2};
   int W = 5;
   knapsack (W, wt, val, n);
   return 0;
}
```

Output DP Table:

```
0  0   0   0   0   0
0  0   12  12  12  12
0  10  12  22  22  22
0  10  12  22  30  32
0  10  15  25  30  37

Maximum value in knapsack = 37
Items included in the knapsack:
Item 4 (Value: 15, weight: 2)
Item 2 (Value: 10, weight: 1)
Item 1 (Value: 12, weight: 2)
```

Prims algorithm.

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

#define MAX 100
#define INF 9999

void prims(int n, int cost[MAX][MAX]){
    int d[MAX], p[MAX], s[MAX];
    int source, min, sum=0;
    int T[MAX][2], k=0;

    min=INF;
    source=0;
    for(int i=0; i<n; i++){.
        for(int( j=0; j<n; j++){
            if(cost[i][j]!=0&& cost[i][j]<min){
                min=cost[i][j];
                source=i;
    }}}
    for(int i=0; i<n; i++){
        s[i]=0;
        d[i]=cost[source][i];
        p[i]=source;
    }
    s[source]=1;

    for(int i=1; i<n; i++){
        min=INF;
        int u=-1;
        for(int j=0; j<n; j++){
```

4/7/24

```
    int(s[j]==0 && d[j]<min){
        min=d[j];
        u=j;
  }}
  T[k][0]=u;
  T[k][1]=p[u];
  k++;
  sum +=cost[u][p[u]];
  s[u]= 1;
  for(int v=0; v<n; v++){
     if(s[v]==0 && cost[u][v]<d[v]){
         d[v]=cost[u][v];
         p[v]=u;
  }}}

  if (sum>=INF){
     printf("Spamming tree does not exist\n");
  } else{
   printf("Spamming tree exists and MST is\n");
     for(int i=0; i<n-1; i++){
        printf("%d-%d\n", T[i][0],
                        T[i][1]);
     }
     printf("The cost of the Min spamming
       Tree is: %d\n", sum);
  }}

int main(){
  int n;
  int cost[MAX][MAX];
  printf("Enter the number of vertices:");
  scanf("%d", &n);
```

```
printf("Enter the cost adjacency, ]\n");
for(int i=0; i<n; i++){
    for(int j=0; j<n; j++){
        scanf("%d", &cost[i][j]);
}}
prims(n, cost);
return 0;
}
```

```
Enter the number of vertices: 6
Enter the cost adjacency matrix:
 0    60   10   9999 9999 9999
60    0    9999 20   40   70
10   9999  0    9999 9999 50
9999 20   9999  0    9999 80
 0   40   9999 9999  0    30
9999 70    50   80   30    0

Spanning tree
2-0
5-2
4-5
1-4
3-1
Cost of MST-150
```