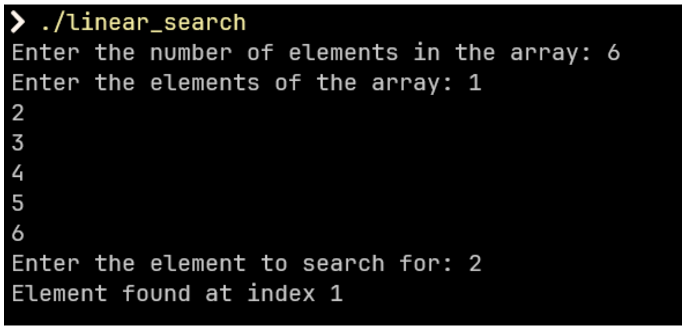
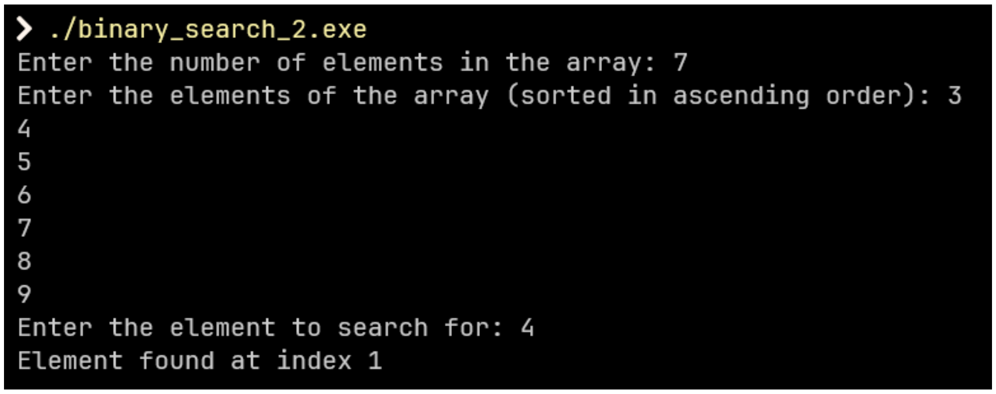
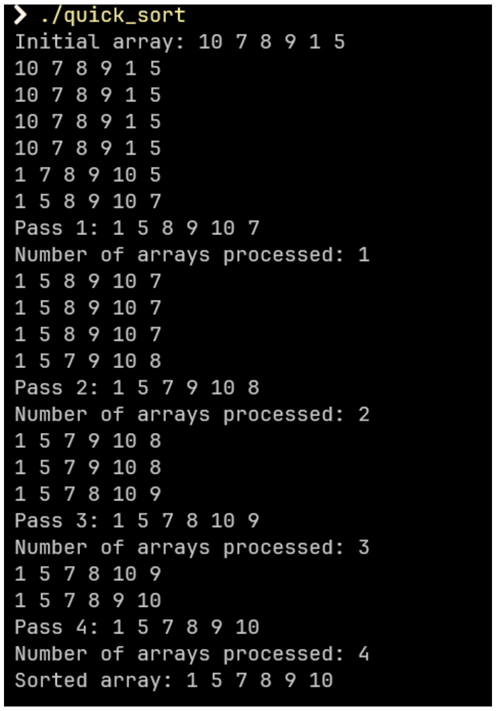


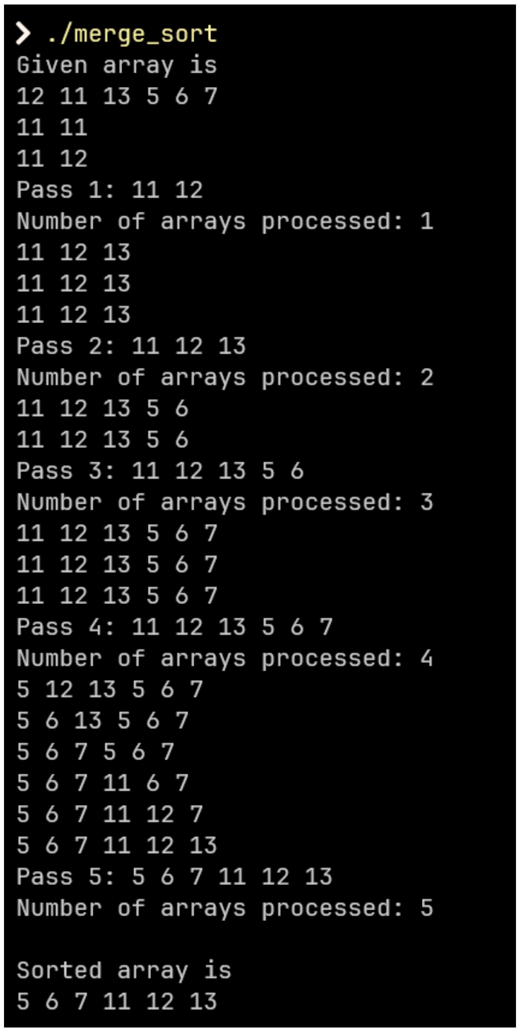
```
❯ ./linear_search
Enter the number of elements in the array: 6
Enter the elements of the array: 1
2
3
4
5
6
Enter the element to search for: 2
Element found at index 1
```



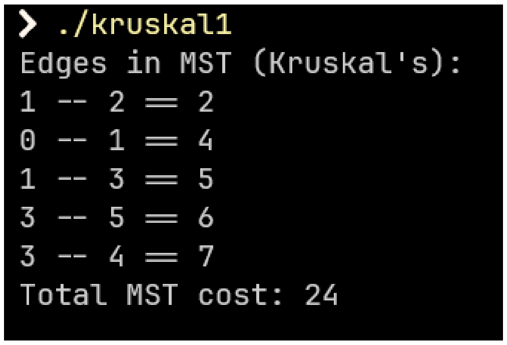
```
❯ ./binary_search_2.exe
Enter the number of elements in the array: 7
Enter the elements of the array (sorted in ascending order): 3
4
5
6
7
8
9
Enter the element to search for: 4
Element found at index 1
```



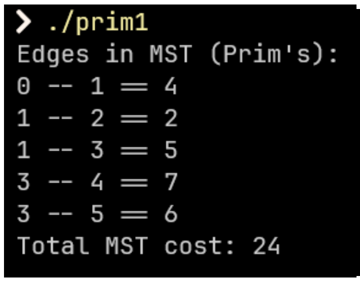
```
❯ ./quick_sort
Initial array: 10 7 8 9 1 5
10 7 8 9 1 5
10 7 8 9 1 5
10 7 8 9 1 5
10 7 8 9 1 5
1 7 8 9 10 5
1 5 8 9 10 7
Pass 1: 1 5 8 9 10 7
Number of arrays processed: 1
1 5 8 9 10 7
1 5 8 9 10 7
1 5 8 9 10 7
1 5 7 9 10 8
Pass 2: 1 5 7 9 10 8
Number of arrays processed: 2
1 5 7 9 10 8
1 5 7 9 10 8
1 5 7 8 10 9
Pass 3: 1 5 7 8 10 9
Number of arrays processed: 3
1 5 7 8 10 9
1 5 7 8 9 10
Pass 4: 1 5 7 8 9 10
Number of arrays processed: 4
Sorted array: 1 5 7 8 9 10
```



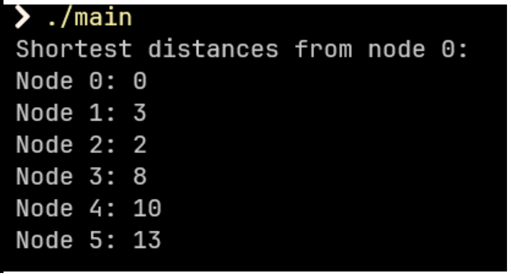
```
❯ ./merge_sort
Given array is
12 11 13 5 6 7
11 11
11 12
Pass 1: 11 12
Number of arrays processed: 1
11 12 13
11 12 13
11 12 13
Pass 2: 11 12 13
Number of arrays processed: 2
11 12 13 5 6
11 12 13 5 6
Pass 3: 11 12 13 5 6
Number of arrays processed: 3
11 12 13 5 6 7
11 12 13 5 6 7
11 12 13 5 6 7
Pass 4: 11 12 13 5 6 7
Number of arrays processed: 4
5 12 13 5 6 7
5 6 13 5 6 7
5 6 7 5 6 7
5 6 7 11 6 7
5 6 7 11 12 7
5 6 7 11 12 13
Pass 5: 5 6 7 11 12 13
Number of arrays processed: 5

Sorted array is
5 6 7 11 12 13
```



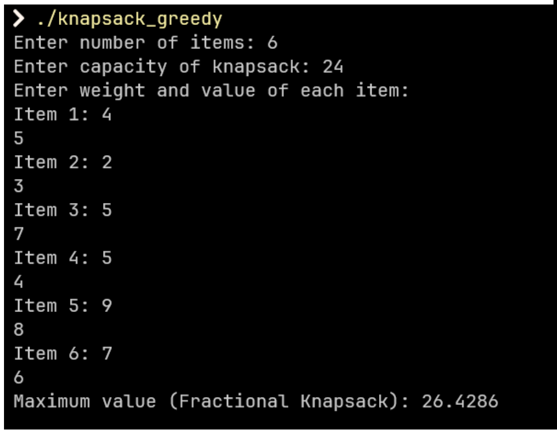
```
❯ ./kruskal1
Edges in MST (Kruskal's):
1 -- 2 == 2
0 -- 1 == 4
1 -- 3 == 5
3 -- 5 == 6
3 -- 4 == 7
Total MST cost: 24
```



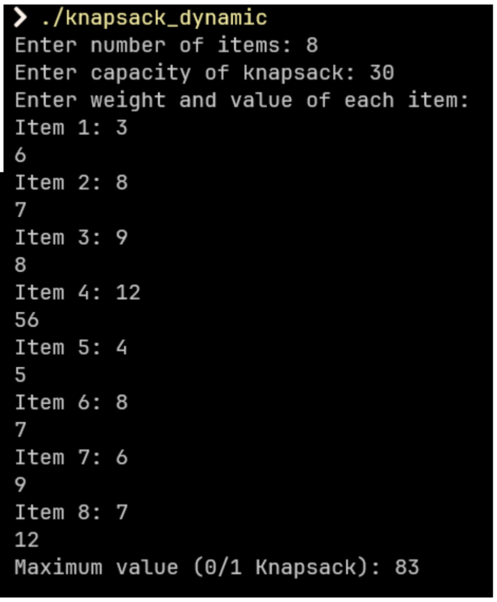
```
❯ ./prim1
Edges in MST (Prim's):
0 -- 1 == 4
1 -- 2 == 2
1 -- 3 == 5
3 -- 4 == 7
3 -- 5 == 6
Total MST cost: 24
```



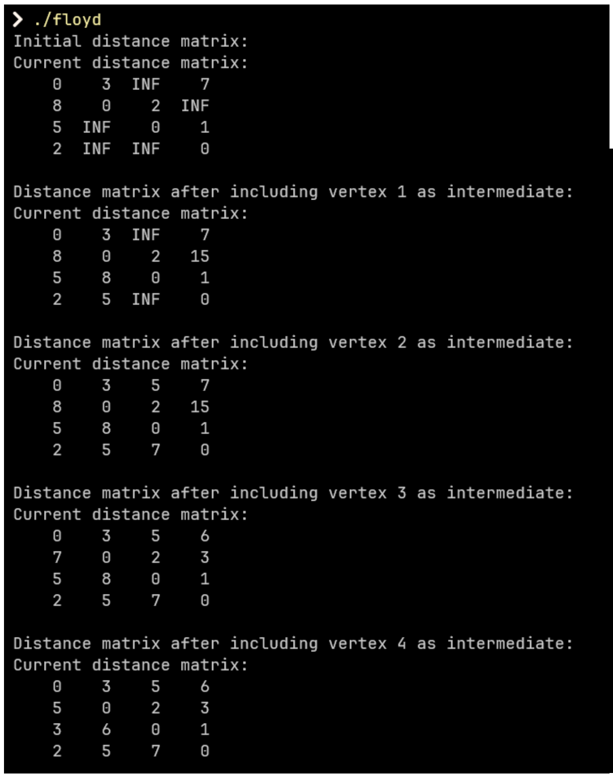
```
❯ ./main
Shortest distances from node 0:
Node 0: 0
Node 1: 3
Node 2: 2
Node 3: 8
Node 4: 10
Node 5: 13
```



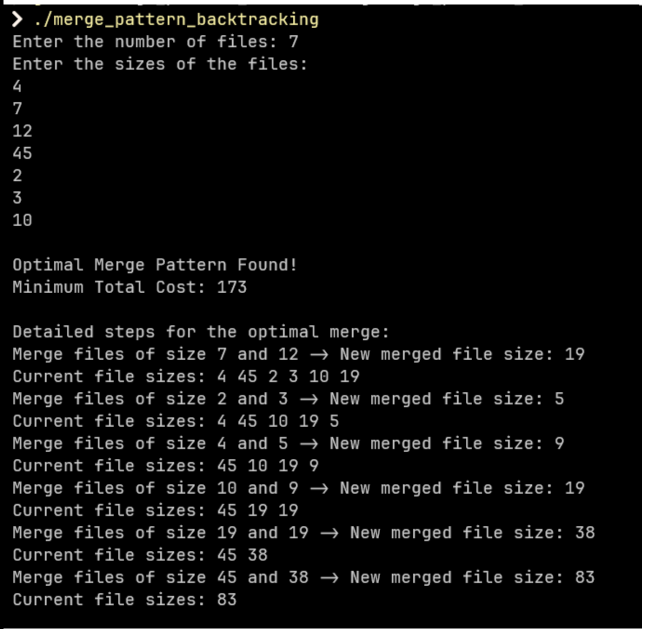
```
> ./knapsack_greedy
Enter number of items: 6
Enter capacity of knapsack: 24
Enter weight and value of each item:
Item 1: 4
5
Item 2: 2
3
Item 3: 5
7
Item 4: 5
4
Item 5: 9
8
Item 6: 7
6
Maximum value (Fractional Knapsack): 26.4286
```



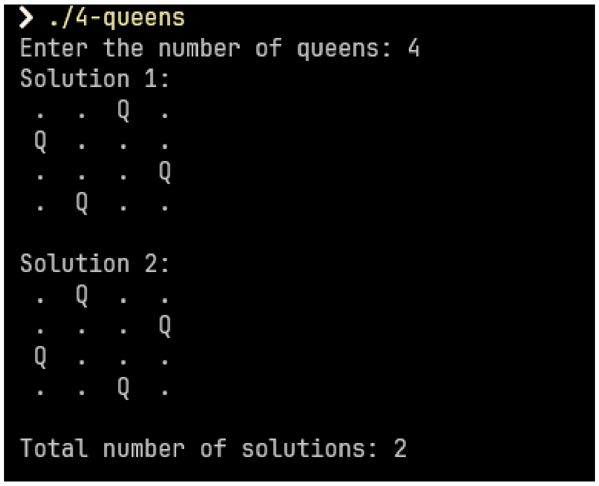
```
> ./knapsack_dynamic
Enter number of items: 8
Enter capacity of knapsack: 30
Enter weight and value of each item:
Item 1: 3
6
Item 2: 8
7
Item 3: 9
8
Item 4: 12
56
Item 5: 4
5
Item 6: 8
7
Item 7: 6
9
Item 8: 7
12
Maximum value (0/1 Knapsack): 83
```



```
> ./floyd
Initial distance matrix:
Current distance matrix:
    0    3  INF    7
    8    0    2  INF
    5  INF    0    1
    2  INF  INF    0

Distance matrix after including vertex 1 as intermediate:
Current distance matrix:
    0    3  INF    7
    8    0    2   15
    5    8    0    1
    2    5  INF    0

Distance matrix after including vertex 2 as intermediate:
Current distance matrix:
    0    3    5    7
    8    0    2   15
    5    8    0    1
    2    5    7    0

Distance matrix after including vertex 3 as intermediate:
Current distance matrix:
    0    3    5    6
    7    0    2    3
    5    8    0    1
    2    5    7    0

Distance matrix after including vertex 4 as intermediate:
Current distance matrix:
    0    3    5    6
    5    0    2    3
    3    6    0    1
    2    5    7    0
```



```
> ./merge_pattern_backtracking
Enter the number of files: 7
Enter the sizes of the files:
4
7
12
45
2
3
10

Optimal Merge Pattern Found!
Minimum Total Cost: 173

Detailed steps for the optimal merge:
Merge files of size 7 and 12 -> New merged file size: 19
Current file sizes: 4 45 2 3 10 19
Merge files of size 2 and 3 -> New merged file size: 5
Current file sizes: 4 45 10 19 5
Merge files of size 4 and 5 -> New merged file size: 9
Current file sizes: 45 10 19 9
Merge files of size 10 and 9 -> New merged file size: 19
Current file sizes: 45 19 19
Merge files of size 19 and 19 -> New merged file size: 38
Current file sizes: 45 38
Merge files of size 45 and 38 -> New merged file size: 83
Current file sizes: 83
```



```
❯ ./4-queens
Enter the number of queens: 4
Solution 1:
 .  .  Q  .
 Q  .  .  .
 .  .  .  Q
 .  Q  .  .

Solution 2:
 .  Q  .  .
 .  .  .  Q
 Q  .  .  .
 .  .  Q  .

Total number of solutions: 2
```