22-2_기말_알고리즘_이현자_002

1. 백트래킹 기법에서 nonpromising 노드는 가지치기를 하여 보다 효율적으로 해답을 찾을 수 있다. N-Queens 문제에서 Promising 함수는 두 개의 퀸이 같은 열에 있는지, 대각선에 있는지를 검사하여 nonpromising 노드인지 아닌지를 결정한다. 배열 Col[i]가 I번째 행에 있는 퀸의 열 위치를 저장하고 있다고 할 때, 다음 N-Queens 알고리즘의 Promising 함수를 완성하시오.

```
boolean promising(index i){
index k = 1;
boolean p = true;
while(k<i && p){
if(_____________________)
p = false;
k++;
}
return p;
```

2. 동적프로그래밍 기법을 사용해 Floyd 최단 경로를 구하는 알고리즘을 완성하시오.

```
void floyd(int n, const number W[][], number D[][]){
    index i, j, k;
    D = W;
        for(k=1; k<=n; k++)
            for(i=1; i<=n; i++)
                for(j=1; j<=n; j++)
                    // 알고리즘을 완성하시오.
                }
}
```

3. Greedy 기법으로 동전 거스름돈 문제를 해결하기 위한 알고리즘을 완성하시오.
단, changes[] 배열에는 액면가가 큰 동전이 큰 순서대로 정렬돼 저장돼있고, amount 변수에는 거스름 돈 액수가 저장, cc[] 배열은 거스름돈에서 필요한 각 동전의 개수 저장하는 용도로 사용.

```
void minChange(int changes[], int amount, int &cc[]){
    cc[]={0};
    for(i=1; i<=r; i++) //r 배열의 크기, 동전 액면가의 가지 수

        // 알고리즘을 완성하시오.

        }
}
```

4. 배낭 빈틈없이 채우기 문제해결을 위한 알고리즘을 완성하시오. 단, 배낭의 용량 M, 아이템의 개수 n, 무게당 이익이 큰 순으로 정렬된 각 아이템의 이득과 무게가 저장된 배열 p[1:n], w[1:n]. x[1:n]는 배낭을 채우는 아이템 항목을 나타내는 배열.

```
void GreedyKnapsack(float M, int n, int p[], int w[], float& x[]){
     for (int i=1; i<=n; i++) x[i] = 0.0; // Initialize x.
     float U = M;
     for (i=1; i<=n; i++) {
       if (w[i] > U) break;
       x[i] = 1.0;
       U -= w[i];
     }
     if (i <= n) { // 알고리즘을 완성하시오. }
}
```

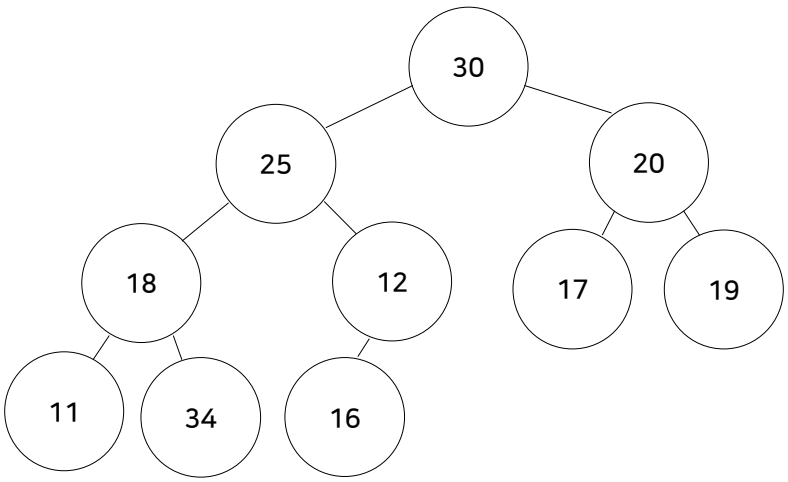
5. 일반적인 되추적 기법 알고리즘을 완성하시오.

```
void checknode(node v){
     if(promising(v))
       if(there is a solution at v)
            write the solution;
       else
            // 알고리즘을 완성하시오.
     }
```

6. 다음 정렬 알고리즘으로 정렬할 때, 출력되는 배열 S의 값을 모두 나타내시오. 단위 연산은 할당이며, 최악의 시간 복잡도를 구하는 과정을 포함하여 답하시오. 배열 S = {55, 97, 3, 6, 41}

```
void Sort(int n, keytype &S[] ){
     index i, j;
     for(i=1; i <= n;i++)
          for (j=1; j <=n-i; j++){
            if (S[ j ] > S[ j + 1 ])
               swap S[ j ] and S[ j + 1 ];
          }
          print S[1....n]
     }
}
```

7. 힙 정렬에서 힙을 만드는 makeheap()에서는 몇 번의 shiftdown이 호출되는가? 이를 답하고, shiftdown 호출 후의 모습을 트리로 그리시오. (과정과 결과를 모두 그리시오)



8. 다음 두 알고리즘의 단위 연산을 할당이라고 할 때, 시간 복잡도를 각각 구하시오.
(구하는 과정 및 식을 포함하시오.)
이때 시간 복잡도는 최악의 시간 복잡도, 모든 경우 시간 복잡도, 평균 시간 복잡도 등 여러 시간 복잡도 중 본인이 할 수 있는 것을 선택하여 작성하면 됨.

```
void Sort(int n, number S[]){
  index i, j;
  for(i = 1; i <= n; i++)
    for(j = i+1; j <= n; j++)
        if(S[j]<S[i])
          swap S[i] and S[j];
}
```

```
void sort(int n, keytype S[]){
  index i, j, smallest;
  for(i=1; i<=n-1; i++){
    smallest = i;
    for(j=i+1; j<=n; j++)
      if(S[j]<S[smallest])
        smallest = j;
    exchange S[i] and S[smallest];
  }
}
```