답지(앞의 숫자는 쪽수)

1-1. 버블정렬은 큰 수부터 순서대로 정렬하는 것으로, 즉 맨 뒤의 이미 큰 수로써 정렬된 것은 정렬할 필요가 없기 때문이다.

2-1. 생략
2-2.

```
n ← length_of_d[]
for i ← 1 to n-1 step 1
  min ← i
  for j ← i+1 to n step 1
    if d[j] < d[min] then
      min ← j
    end if
  end for
  swap(d[i], d[min])
end for
```

3-1. 생략
3-2. (이 내용은 교과서를 기준으로 작성됨)

```
n ← length_of_d[]
for i ← 2 to n step 1
  j ← i
  while j > 1 and d[j-1] > d[j]
    swap(d[j], d[j-1])
    j ← j - 1
  end while
end for
```

5-1. 생략
5-2. 생략

7-1. 생략
7-2. 생략
7-3.

```
  if(k > m/2) return;
  int i=2*k;
  if(d[i+1]>d[i]&&i<m) i++;
  if(d[k]<d[i]){}
```

7-4. $O(n\lg n)$
7-5. 생략

9-1. 생략
9-2. merge함수: 묶음끼리를 비교하여 두 묶음을 오름차순으로 한 묶음으로 병합
mergesort함수: 자료를 절반씩 나누어가며 한 개의 자료로 나눔, merge 부름

```
  for(int k=l; k<=r; k++)
    if(i<=m && (d[i]<=d[j] || j>r))
      temp[k] = d[i++];
    else
      temp[k] = d[j++];
  for(int k=l; k<=r; k++)
    d[k] = temp[k];
```

11-1. 이진탐색. 과정 생략
11-2. $O(n)$, $O(\lg n)$
11-3. 생략
11-4. 정렬 후 이진탐색을 실행하면 됨. 코드 생략

13-1. 1, 2, 3, 4, 5, 7, 6, 8, 9, 10, 0
13-2.

```
  printf("%d ", t);
  v[t]=1;
  for(int i=1;i<=n;i++)
    if(g[t][i]==1&&v[i]==0)
      DFS(i);
```

15-1. 1, 2, 5, 3, 4, 7, 9, 6, 8, 10, 0
15-2.

```
int q[100], rear, front=1;
bool q_empty(){
    if(rear<front) return true;
    else return false;
}
void q_push(int s){ q[++rear]=s; }
void q_pop(){ front++; }
int q_top(){ return q[front]; }
```

16-1. 계산문제: 1, 5, 6, 7

16-2. ① (연산 문제가 아니라 모든 문제)
16-3. 결정문제

18-1. ① ② ③ ④
18-2. 생략

21-1.

알고리즘	최선시간복잡도	평균시간복잡도	최악시간복잡도
버블 정렬	$\Omega(n^2)$	$\Theta(n^2)$	$O(n^2)$
선택 정렬	$\Omega(n^2)$	$\Theta(n^2)$	$O(n^2)$
삽입 정렬	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$
퀵 정렬	$\Omega(n\log n)$	$\Theta(n\log n)$	$O(n^2)$
힙 정렬	$\Omega(n\log n)$	$\Theta(n\log n)$	$O(n\log n)$
병합 정렬	$\Omega(n\log n)$	$\Theta(n\log n)$	$O(n\log n)$
순차 탐색	$\Omega(1)$	$\Theta(n)$	$O(n)$
2진 탐쌕	$\Omega(1)$	$\Theta(\log n)$	$O(\log n)$

21-2.

버블·삽입·선택 정렬	$O(n^2)$	퀵·힙 정렬	$O(n\lg n)$

21-3.
$O(n)$, $O(1)$,, 따라서 오른쪽이 더 효율적인 알고리즘이다.
21-4. $O(n)$, $O(n^2)$, $O(\log n)$, $O(n!)$, $O(n^2!)$, $O(2^n)$

22-1. 순차 탐색: 전체 탐색 알고리즘, 2진 탐색: 부분 탐색 알고리즘
22-2.

탐색 공간의 구조	탐색 알고리즘
선형 구조	순차 탐색 2진 탐색
비선형 구조	깊이 우선 탐색 너비 우선 탐색

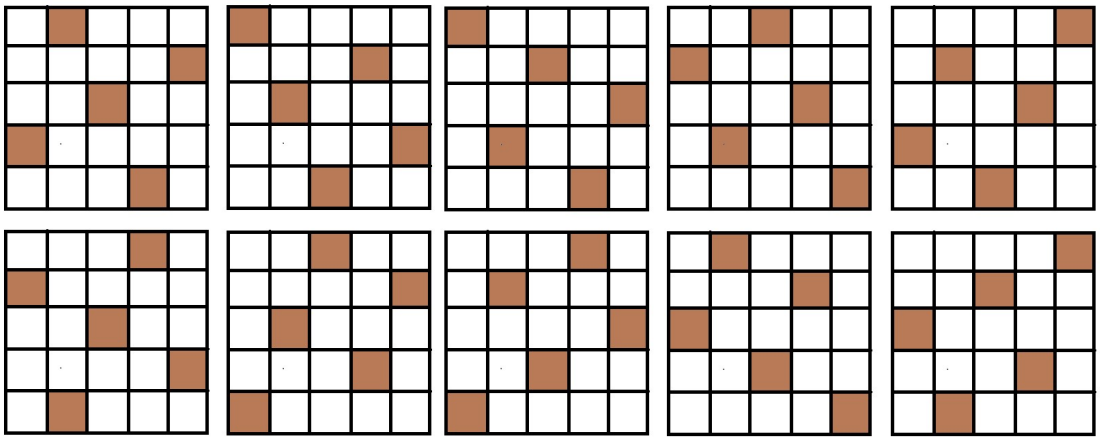
22-3.
- 선형 자료인 리스트나 배열에서 원하는 자료를 찾는 것
- DB 등에서 사용자가 원하는 자료를 찾기 위해 검색하는 것
- 비선형 자료구조인 트리 또는 그래프 등에서 원하는 자료를 찾는 행위
- 주어진 문제를 해결하는 과정 중 문제 공간에서 해를 찾는 것
- 탐색한 해의 개수를 세는 계수 문제
- 탐색한 해 중 가장 품질이 좋은 해를 찾는 최적화 문제

그 외에도 의미만 맞으면 정답

25-1. $\{1,2,3,4,5\}$
$\{\{1,1,1\},\{1,1,2\},\{1,1,3\},\{1,1,4\},\{1,1,5\},\{1,2,2\},\{1,2,3\},\{1,2,4\},\{1,2,5\},\cdots,\{5,5,5\}\}$
25-2. 4-3-8 또는 4-5-6 (합 15)

25-3. 70 (59, 2, 9)

25-4.



27-1. 1, 2, 3, 4, 5, 7, 8

27-2. 같은 문제: 교과서 232쪽 탐구 03. 문제 이해가 안 되면 해당 문제를 참고

탐색공간: {0개 0개 0개 0개}, {0개 0개 0개 1개}, {0개 0개 0개 2개} ..... 등 가능한 모든 경우수 → 앞에서부터 순서대로, {0개, , , }, {1개, , , } // {1개, 0개, , }, {1개, 1개, , }, {1개, 2개, , } // {1개, 2개, 0개, } ..... 으로 탐색

```
#include <stdio.h>

int main(){
        int cnt=0, n;
        scanf("%d", &n);
        cnt+=n/500;  n%=500;
        cnt+=n/100;  n%=100;
        cnt+=n/50;  n%=50;
        cnt+=n/10;  n%=10;

        printf("%d", cnt);

        return 0;
}
```