알고리즘

- 일반적으로 어떤 문제를 해결하는 절차를 말함.
- 입력이 존재하고 입력에 대한 출력이 존재하고, 언젠가는 끝나야 한다.
- 같은 입력에 대해 빠른 해결 시간을 갖는 효율적인 알고리즘이 중요한다.
- 알고리즘 학습은 이미 잘 알려진 문제에 대한 해결 방법을 배워서 활용하기 위한 것이다.

- 알고리즘 학습 단계
  - 1단계 : 알고리즘 학습을 위한 배열 연습.
  - 2단계 : 재귀, 그래프 탐색, 백트래킹.
    - 재귀호출, 조합, 순열, DFS, BFS, 위상정렬
    - 미로 찾기 등
  - 3단계 : 그래프 최적화, 동적계획법(DP)
    - 프림, 크루스컬, 다익스트라, DP
    - 최단거리, MST, 연속행렬 곱셈 등
- 일반적인 알고리즘 교재는 문제풀이 중심.
- 수식과 관련된 내용은 이산수학에서 다룸.

데이터 입출력

- scanf(), printf() 사용이 편리.
- 디버깅을 위해 파일에서 입력하기.

```
// main()의 가장 앞부분에 추가. 서버 제출시 삭제.
freopen(“input.txt“, “r”, stdin);
```

```
// Java의 경우
System.setIn(new FileInputStream("input.txt"));
Scanner sc = new Scanner(System.in);
```

자료형

- 정수형
  - 연산결과는 64-bit 정수형이 필요한 경우가 있음.
    - long long (C언어), long (Java)
  - 64-bit를 넘어가는 정수형을 다루는 경우 수식으로 처리해야 함.
  - C의 경우 scanf()의 서식문자로 입력을 한 자리 씩 읽을 수 있음.
    - 예) %1d, %1x
  - for문에는 int형 사용.
  - 대략 20억 이내까지 저장 가능.

- 실수형
  - double형 사용 (float은 표현할 수 있는 범위가 작음).
  - 고정형 sign bit를 사용하므로 unsinged를 붙여도 자리수가 늘어나지 않음.
  - 출력에 서식문자를 사용하면 마지막 자리 뒤에서 반올림 됨.
    - 예) %.3f
  - 대략 유효 자리수 13자리 정도 저장.

배열

- 배열의 크기에 주의해야 함.
  - 인덱스를 0번부터 사용하는 경우와 1번부터 사용하는 경우.
- 인덱스 연산은 범위를 미리 정의하고 코딩을 시작해야 함.
  - 배열의 끝 부분을 읽지 않는 경우가 많으므로 주의.
- C언어의 경우 큰 배열은 전역으로 선언한다.
  - 스택 영역의 크기가 제한되어 있기 때문
- 배열의 최대 크기.
  - 정수형 백만개를 저장하는 1차원 배열(혹은 1000x1000 2차원 배열).
  - 배열에 저장할 수 없는 경우 수식을 알아내야 함.

- 배열 인덱스
  - 1개의 인덱스 변수로 배열에 접근하기

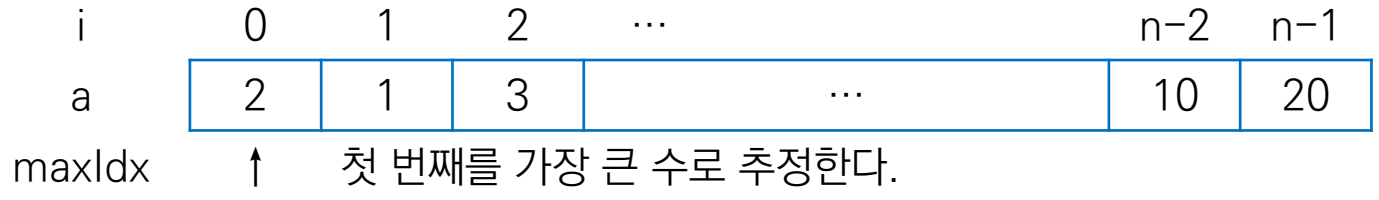
배열을 벗어나지 않도록 주의한다.

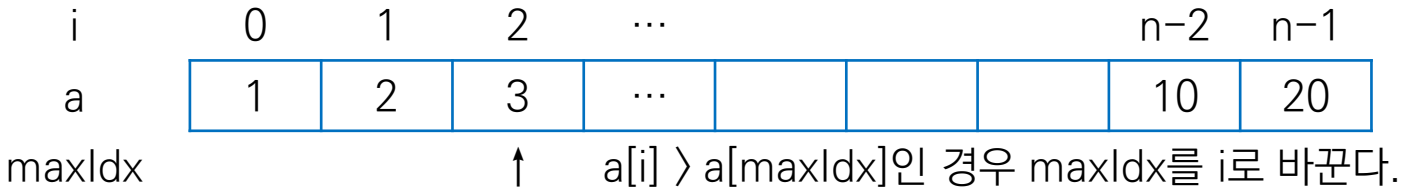
```
for( int i = 0 ; i < n ; i++)
{

}
```

가장 큰 수 찾기

- 2개의 인덱스 변수로 배열에 접근하기






```
maxIdx = 0;
for( i = 1 ; i 〈 n ; i++ )
{
      if( a[i] 〉 a[maxIdx] )
            max = i;
}
```

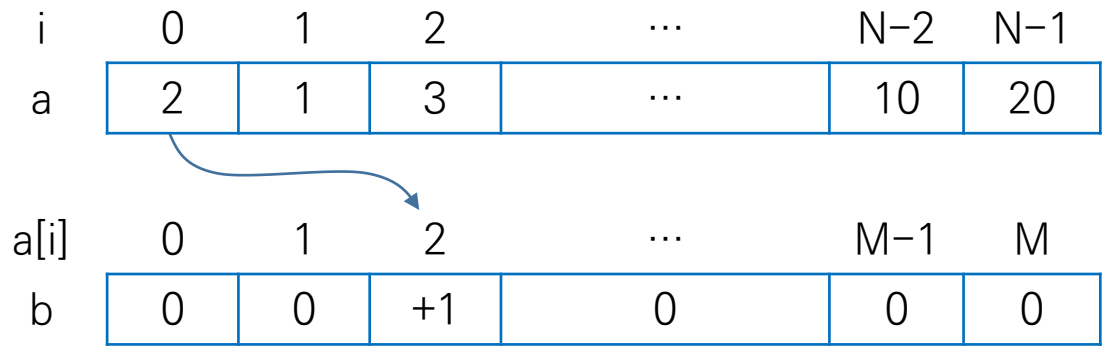
연습문제

- N개의 정수가 들어 있는 배열 A[i]에서 가장 큰 수와 가장 작은 수의 차이를 출력하시오.
- 5 <= N <= 1000
- 10개 이내의 테스트케이스가 주어진다.

입력
1 // 테스트케이스 개수 5 // N 477162 658880 751280 927930 297191
출력
#1 630739

같은 원소의 개수 구하기

- 다른 배열의 정보를 기록하는 배열
  - a[i]에 저장된 원소를 다른 배열의 인덱스로 사용한다. 0<=a[i]<=M
  - 0으로 초기화한 배열에 인덱스가 가리키는 위치의 값을 하나 증가한다.
  - 모든 a[i]에 대해 반복 한다.




```
for( i = 0 ; i < N ; i++ )
{
    b[a[i]]++;
}
```

연습문제

- 0부터 9까지 숫자가 적힌 N장의 카드가 주어진다. 숫자별로 카드가 몇 장 인지 세어, 가장 많은 카드에 적힌 숫자와 카드 장수를 출력하시오. 단, 카드 개수가 같을 때는 적힌 숫자가 큰 쪽을 출력한다.
- 5 <= N <= 100

입력
1 // 테스트케이스 개수 5 49679 // N과 한 칸의 공백, 카드 숫자
출력
#1 9 2 // 테스트케이스 번호, 빈칸, 카드에 적힌 숫자, 빈칸, 카드 수

- 공백 없는 숫자 입력
  - '0에서 9까지의 숫자가 적힌 5장의 카드'의 예

```
int N;
scanf("%d", &N);

for(int i = 0; i < N; i++)
{
    scanf("%1d", &a[i]);
}
```

Text.txt

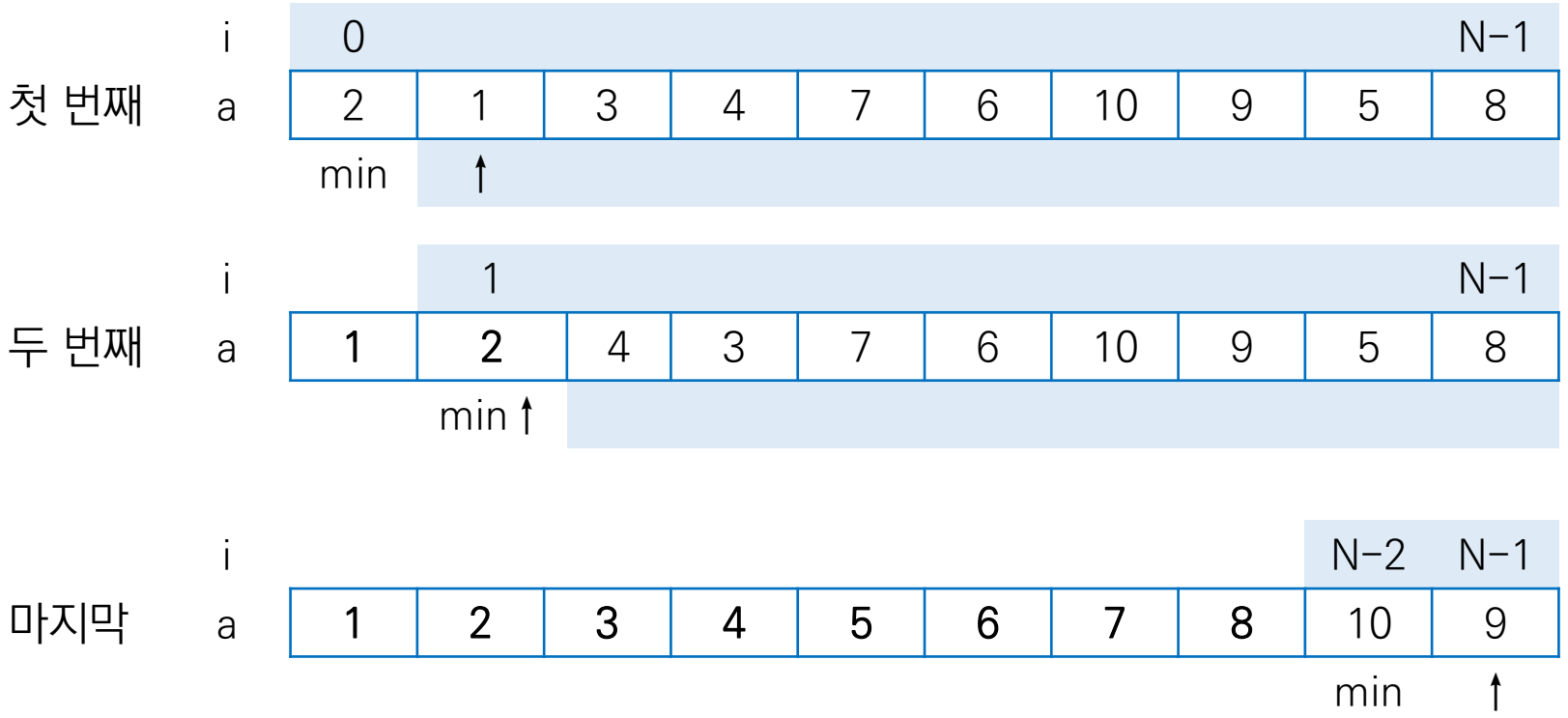
5 49679

```
int N = sc.nextInt();
String str = sc.next();

for(int i = 0; i < N; i++)
{
    a[i] = str.charAt(i) – '0';
}
```

배열의 활용

- 선택 정렬 (오름차순)
  - 크기가 N인 배열의 각 구간에서 맨 처음 값을 최소로 추측한다.
  - 나머지 요소와 비교해 실제 최소를 찾는다.



- 필요한 정보

n개의 원소인 경우
각 구간의 시작을 나타내는 i : 0 -> n-2
각 구간 min의 초기값 : a[i]
각 구간에서의 min과의 비교 위치 j : i+1 -> n-1

인덱스에 대한 정보를 먼저 확인하고 코드를 작성한다.

```
for ( i = 0 ; i < n-1 ; i++ )
{
    minIdx = i;
    for( j = i+1; j < n ; j++ )
    {
        if( a[minIdx] > a[j] )
            minIdx = j;
    }
    temp = a[i];
    a[i] = a[minIdx];
    a[minIdx] = temp;
}
```

거품 정렬과 비교해보세요.

연습문제

- N개의 정수를 가장 큰 수와 가장 작은 수가 번갈아가며 나타나도록 정렬하고, 10개까지 출력하시오.
  - 입력) 1 2 3 4 5 6 7 8 9 10
  - 출력) 10 1 9 2 8 3 7 4 6 5
- 10 <= N <= 100, 1에서 100사이의 정수가 주어짐.

입력
1 // 테스트케이스 개수 10 1 2 3 4 5 6 7 8 9 10 // N과 N개의 정수
출력
#1 10 1 9 2 8 3 7 4 6 5

- 배열원소를 3개씩 읽어서 합 구하기.

연습문제 – 1차원 배열의 부분합

- N개의 정수가 들어있는 배열에서 이웃한 원소 M개씩의 합을 구하고, 가장 큰 경우와 가장 작은 경우의 차이를 출력하시오.
- 10 〈= N 〈= 100, N 〉 M,
- 2 〈= M 〈 N;
- 1에서 10000사이의 정수가 주어짐.

입력
1 // 테스트케이스 개수 10 3 // N M 1 2 3 4 5 6 7 8 9 10
출력
#1 21

연습문제 – 문자열 비교

- 길이가 N, M인 패턴과 텍스트 두 개의 문자열이 주어진다.
  - 패턴 DEF, 텍스트 ABCDEFG –〉 텍스트 안에 패턴이 존재
- 텍스트에서 패턴과 일치하는 부분이 있으면 1, 없으면 0을 출력하시오.
- 10 〈= M 〈= 1000, 5 〈= N 〈= 100

입력
1 // 테스트케이스 개수 XYPV // 패턴 EOGGXYPVSY // 텍스트
출력
#1 1

연습문제 – 16진수 –〉 2진수 바꾸기

- 길이가 N인 16진수 한자리를 4자리 2진수로 바꿔 출력하시오.
- 4 〈= N 〈= 10

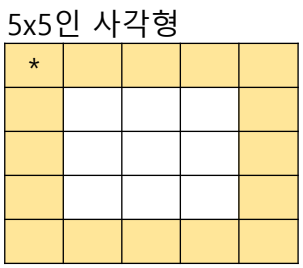
입력
1 // 테스트케이스 개수 47FE // 16진수
출력
#1 0100011111111110

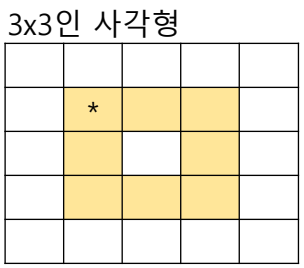
- 10진수 4를 비트 단위로 표시해보면…

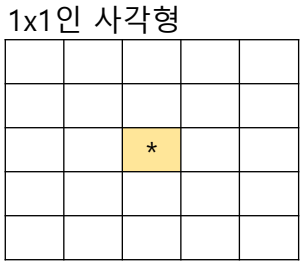
10진수	b3	b2	b1	b0
4	0	1	0	0

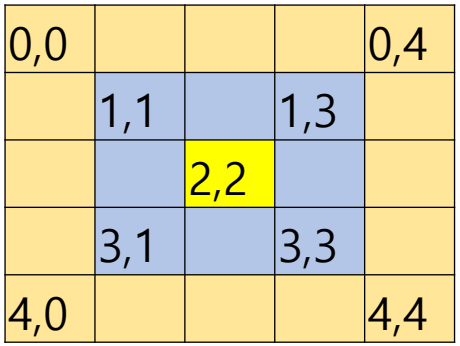
- 비트 연산을 이용하거나 2차원 배열에 2진수 표현을 저장.

- 접근 방법









배열의 크기가 n x n이고,
사각형의 왼쪽 위 좌표가
a[i][i]라고 하면,
0 <= i <= n/2

연습

- 크기가 NxN인 2차원 배열에 다음과 같은 모양으로 숫자가 저장된다.
- 크기N과 행, 열이 주어지면 해당 위치에 저장된 숫자를 출력하시오.

	0	1	2	3	4	5	열
0		1	6	10	13	15	
1			2	7	11	14	
2				3	8	12	
3					4	9	
4						5	
5							
행							

입력
1 // 테스트케이스 개수 6 2 4// N 행 열
출력
#1 8

- 접근 방법

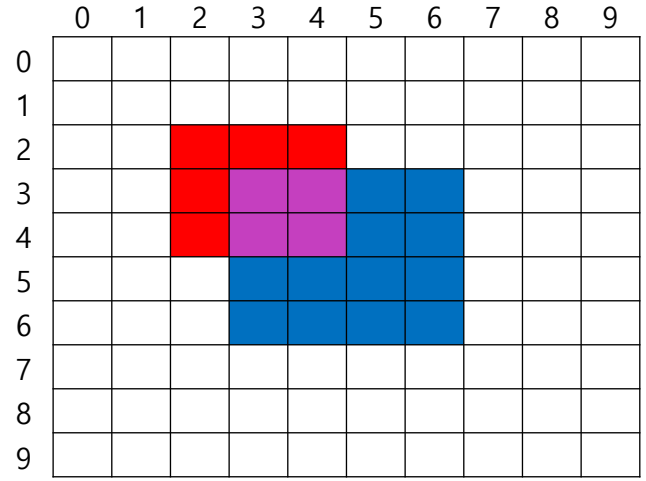
인덱스로 표시한 접근 순서

1	2	3	4	5	→행과 열의 차이 i
0, 1 (1)	0, 2 (6)	0, 3 (10)	0, 4 (13)	0, 5 (15)	
1, 2 (2)	1, 3 (7)	1, 4 (11)	1, 5 (14)		
2, 3 (3)	2, 4 (8)	2, 5 (12)			
3, 4 (4)	3, 5 (9)				
4, 5 (5)					
4	3	2	1	0	마지막 행 번호 N-i

```
k = 1;
for (i = 1; i <= N; i++)   // 행과 열의 차이
{
   for (j = 0; j <= N − i; j++)  // 행 번호
   {
      arr[j][j + i] = k++;  // 열 번호 j+i
   }
}
```

연습문제 – 색칠하기

- 크기가 10x10인 2차원 배열에 빨강색과 파란색을 칠하려고 한다. 이때 겹치는 부분은 보라색이 된다. 보라색으로 칠해진 칸 수를 출력하시오.
- 테스트 케이스 별로 칠하는 영역의 왼쪽 모서리, 오른쪽 아래 모서리, 색상(1빨강, 2 파랑)이 주어진다.



입력
1 // 테스트케이스 개수 2 // 칠하는 횟수 2 2 4 4 1 // 왼쪽 위, 오른쪽 아래, 색상 3 3 6 6 2
출력
#1 4

- 접근 방법
  - 빨간색 영역은 배열에 1을 더한다.
  - 파란색 영역은 배열에 2를 더한다.
  - 값을 더할 때 3이 되는 칸 수를 세서 출력한다.

연습문제 – 사각형 찾기

- 크기가 NxN인 2차원 배열 내부에 1로 채워진 사각형이 있다. 사각형의 가로 세로 칸수를 곱한 값을 출력하라. 사각형이 여러 개인 경우 곱이 가장 큰 경우를 출력하라.

입력	출력
10 10 0000000000 0000000000 0000000000 0001100000 0001100000 0001100000 0001100000 0001100000 0001100000 0000000000	#1 12

- 접근 방법
  - 사각형의 왼쪽 위 모서리를 찾는다.
  - 가로로 0이 나올 때 까지 칸수를 센다.
  - 세로로 0이 나올 때 까지 칸수를 센다.
  - 크기를 알아낸 사각형은 지운다.
  - 새로운 사각형의 크기가 더 크면 이전에 구한 값을 새 값으로 바꾼다.

연습문제 – 2차원 배열의 부분합

- 크기가 NxN인 배열의 부분 배열인 nxm 배열의 합 중 가장 큰 값을 출력하시오.
- 10<=N<=100, 1<=n, m<N

입력	출력
1 //테스트케이스 개수 10 4 6 // N n m 1 0 0 1 0 0 0 0 0 1 0 1 1 1 0 1 0 1 0 1 0 1 1 1 0 0 0 1 0 0 0 1 1 0 0 1 1 1 0 0 1 0 1 1 1 0 0 0 0 1 1 1 0 1 0 1 0 1 1 1 1 0 0 1 1 0 1 1 0 1 0 1 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 0 0	#1 12

- 접근 방법
  - 배열을 읽을 때 가장자리를 벗어나지 않도록 주의한다.
  - nxm배열의 인덱스를 계산하기 어려우면 3x3 정도로 바꿔서 생각한다.
  - 5x5 배열에서 nxm이 2x3인 크기로 읽는다면, 맨 마지막 2x3 배열의 시작 위치는 A[5-n][5-m]이 된다.