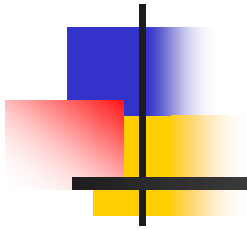
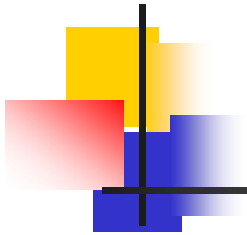


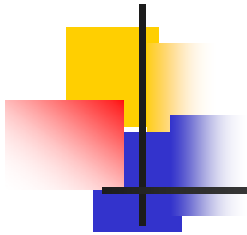
마지막 강의

동적 프로그래밍



목차

- 동적 프로그래밍 소개
- 동적 프로그래밍의 예
- 실습
- 스스로 프로그래밍
- 첨삭 지도

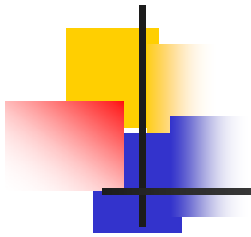


1. 동적 프로그래밍 소개

- 분할정복법
  - 하향식 해결법
  - 나누어진 부분들 사이에 서로 상관관계가 없는 문제를 해결하는데 적합
  - 상관 관계가 있을 경우 비효율적일 수 있음

- 동적 프로그래밍
  - 상향식 해결법
  - 분할정복법과 마찬가지로 문제를 나눈 후에 나누어진 부분들을 먼저 해결
  - 해결된 부분의 결과를 저장한 다음, 후에 그 결과가 필요할 때마다 저장된 값을 이용

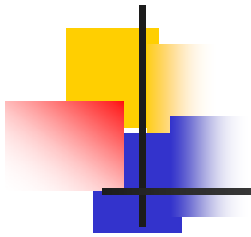
    ↔ 분할정복법(매번 계산)



동적 프로그래밍의 접근 방법

- 개요
  - 주어진 문제를 여러 개의 작은 문제로 분할
  - 작은 문제들의 해를 먼저 구하여 저장
  - 더 큰 문제의 해를 구할 때 작은 문제의 해를 사용
  - 핵심

    작은 문제의 해로부터 큰 문제의 해를 구하는 관계식을 설정



분할 정복법과 동적 프로그래밍

분할정복법(DC)

```
int fib(int n)
{
    if (n <= 1)
        return n;
    else
        return fib(n-1) + fib(n-2);
}
```

fib(10)을 호출할 경우, fib(8)이 2번 실행

동적 프로그래밍(DP)

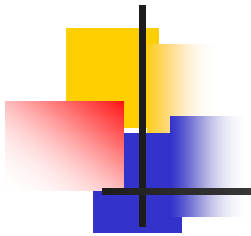
```
int fib2(int n)
{
    index i;
    int f[0..n];

    if (n <= 1)     return n;
    f[0] = 0; f[1] = 1;
    for (i = 2; i <= n; i++)
            f[i] = f[i-1]+f[i-2];
    return f[n];
}
```

fib2(10)을 호출할 경우, fib2(8)의 결과는 f[8]에 이미 저장.

동적 프로그래밍의 특징과 장단점

- 특징
  - 주어진 문제의 해를 구하기 위한 재귀 관계식(Recursive Property)을 구성하여야 한다.
  - 상향식으로 작은 부분 문제부터 해를 구하여야 한다.
- 장점
  - 프로그램을 구현할 때에는 필요한 모든 가능성을 고려해서 구현하게 된다. 따라서 동적 프로그래밍을 이용하여 항상 최적의 결과를 얻을 수 있다.
- 단점
  - 모든 가능성에 대한 고려가 불충분할 경우 최적의 결과를 보장할 수 없다. 동적 계획법을 구현하기 위해서는 충분히 많은 가능성에 대한 고려를 해야 한다.
  - 다른 방법론에 비해 많은 표(배열)을 이용하므로 메모리를 많이 요구한다.



2. 동적 프로그래밍의 예

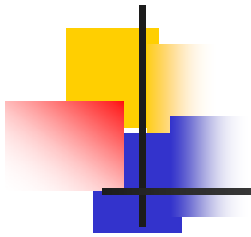
- 이항 계수 구하기
- 최대 합 구하기
  - 변형: 행렬 경로 문제

2.1 이항 계수 구하기

$$\binom{n}{k} = \begin{cases} \binom{n-1}{k-1} + \binom{n-1}{k} & \text{if } 0 < k < n \\ 1 & \text{if } k = 0 \text{ or } k = n \end{cases}$$

분할 정복법을 이용한 알고리즘

```
int bin(int n, int k) {
      if (k == 0 || n == k)
         return 1;
      else
         return bin(n-1,k-1) + bin(n-1,k);
}
```



이항 계수 – 분할 정복법(1)

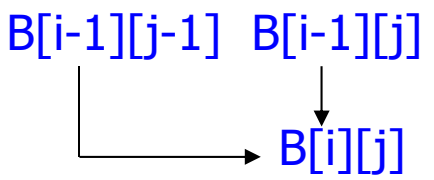
- 재귀 관계식의 구성
  - bin(n, k) = bin(n-1, k-1) + bin(n-1, k)
    - bin(n-1, k-1)과 bin(n-1, k)을 먼저 계산해서 저장
    - 어디에 저장? 이차원 배열이 필요할 것 같다!

- B[i][j] = B[i-1][j-1] + B[i-1][j]
  - 우리가 구하는 최종 해: B[n][k]
  - 반복문의 형태로 i와 j를 증가
    - i값을 0부터 n까지 증가
    - j값도 0부터 k까지 증가

이항 계수 – 분할 정복법(2)

- $\binom{n}{k}$를 구하기 위해서는 아래와 같이 B[0][0]부터 시작하여 위에서 아래로 재귀 관계식을 적용하여 배열을 채워 나가면 된다. 결국 값은 B[n][k] 에 저장된다.

	0	1	2	3	4		j		k
0	1								
1	1	1							
2	1	2	1						
3	1	3	3	1					
4	1	4	6	4	1				
i									
n									

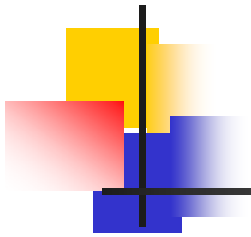


이항 계수 – 분할 정복법(3)

- 문제: 이항계수를 계산한다.
- 입력: 음수가 아닌 정수 n과 k, 여기서 k ≤ n
- 출력: bin, 이항계수

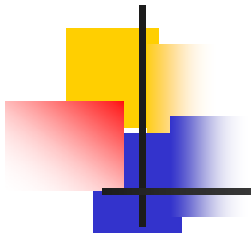
- 알고리즘:

```
int bin2(int n, int k) {
        index i, j;
        int B[0..n][0..k];
        for(i=0; i <= n; i++)
                for(j=0; j <= minimum(i, k); j++)
                        if (j==0 || j == i)
                                B[i][j] = 1;
                        else B[i][j] = B[i-1][j-1] + B[i-1][j];
        return B[n][k];
}
```



2.2 합이 최대인 부분 리스트 구하기

- 문제 설명
  - 입력: 실수 배열 A[n]에 저장된 리스트
    - 예: 2.3 3.2 -4.5 2.1 -5.3 3.6 4.1 -2.3 3.5 -4.5
  - 출력: 연속된 수들의 합이 최대인 리스트를 찾아, 그 합과 리스트를 출력
    - 예: 합 = 8.9, 리스트 = 3.6 4.1 -2.3 3.5



해결 방법

- 어떻게 풀 것인가?
  - A[i]에서의 최대 합을 B[i]에 저장하자.
  - B[i]를 계산하기 위해서는 B[i-1]이 무엇인지 알아야지.
    - B[i-1] > 0, 리스트 연장 (B[i] = B[i-1] + A[i])
    - B[i-1] <=0, 리스트를 새로 시작 (B[i] = A[i])
  - i를 0부터 n까지 증가하는 반복문으로 구현
    - B[0] = A[0]로 초기화
  - 이렇게 계산된 B[] 중에서 제일 큰 값이 해
  - 리스트의 출력 방법?

알고리즘

```
int maxSubList(double A[], int n) {
    double B[n], max;

    max = B[0] = A[0];
    for (int i = 1; i < n; i++) {
        B[i] = (B[i-1] <= 0) ? A[i] : B[i-1] + A[i];
        if (max < B[i])
            max = B[i];
    }
    return max;
}
```

리스트도 출력할 수 있도록 수정해볼 것!

동적 프로그래밍 실습 – 1

- 양의 정수 **n**과 **k**를 입력받아 분할 정복법과 동적 프로그래밍으로 이항 계수를 각각 계산하여 출력하라.

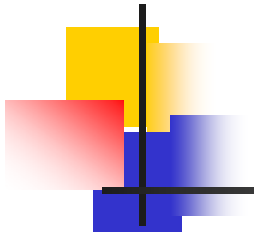
- 두 프로그램의 실행 시간을 비교해보라.

```
#include <time.h>

clock_t start, stop;        // 시작 시간과 종료 시간
double duration;            // 경과 시간 = stop – start

start = clock();            // 시작 시간 측정
bin(n, k);                  // bin 함수 호출
stop = clock();             // 종료 시간 측정

duration = ((double) (stop – start)) / CLOCKS_PER_SEC;
```



동적 프로그래밍 실습 – 2

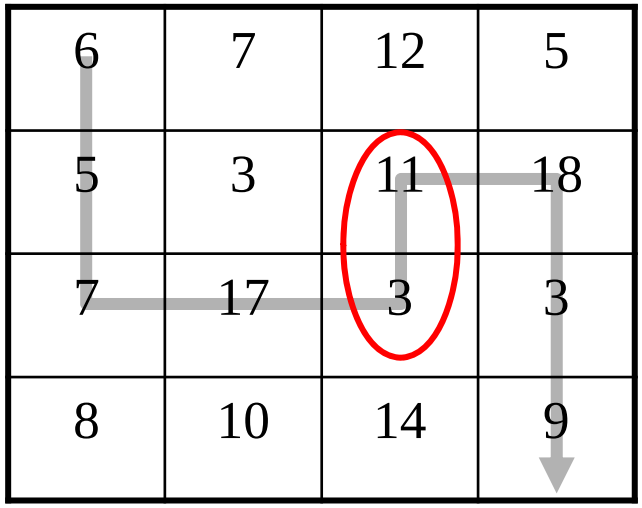
- **"합이 최대인 부분 리스트 구하기" 알고리즘을 완성하라.**
  - 부분 리스트를 출력할 수 있도록 할 것

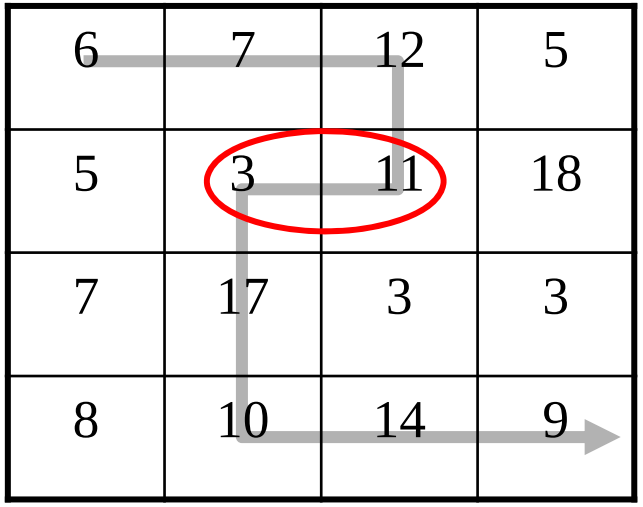
- 무작위로 **-100**에서 **100**사이의 실수를 **20**개 생성하여 **A[50]**에 저장하라.
  - 배열 내용을 먼저 출력하고,
  - 위 알고리즘을 적용한 결과 (부분 리스트의 합, 부분 리스트)를 출력하라.

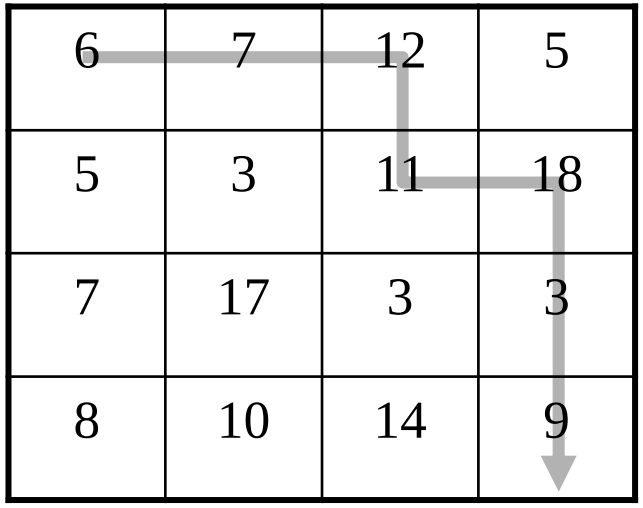
동적 프로그래밍 실습 – 3

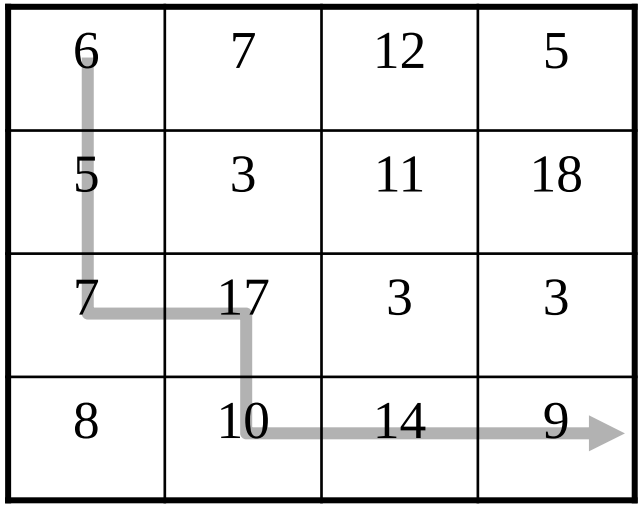
- 행렬 경로 문제
  - 규칙
    - int A[n][n]행렬이 주어지고, A[0][0]에서 시작하여 A[n-1][n-1]까지 이동
    - 현 위치에서 오른쪽이나 아래쪽으로만 이동할 수 있으며, 왼쪽, 위쪽, 대각선 이동은 허용하지 않는다
  - 목표
    - A[0][0]에서 A[n-1][n-1]까지 이동하되, 방문한 칸에 있는 수들을 더한 값이 최대가 되도록 경로 설정
  - 출력
    - 최대 경로의 합
    - 경로 출력 방법도 고려해 볼 것! (option)

이동 방법의 예


6 7 12 5
5 3 11 18
7 17 3 3
8 10 14 9


6 7 12 5
5 3 11 18
7 17 3 3
8 10 14 9


6 7 12 5
5 3 11 18
7 17 3 3
8 10 14 9


6 7 12 5
5 3 11 18
7 17 3 3
8 10 14 9