Problem Solving Techniques 문제해결

Jinkyu Lee

Dept. of Computer Science and Engineering,
Sungkyunkwan University (SKKU)

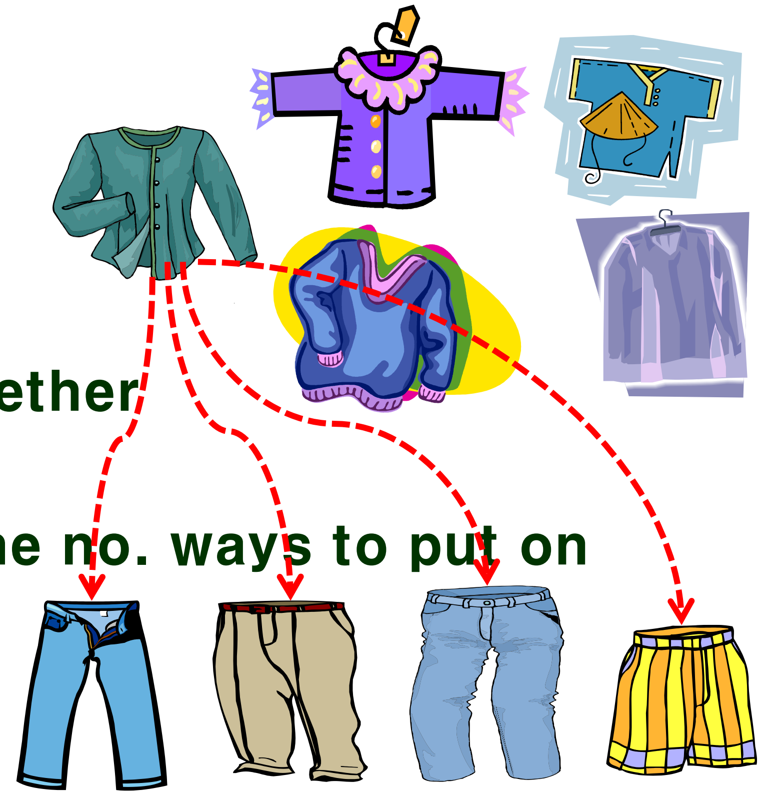
Contents

- Chapter 6 – Combinatorics

1. Basic Counting <1>

- **What is Combinatorics?**
  - Math. Notion on Counting

- **Rule of Product**
  - #Cases that A & B occur together
  - If $|A|=m$, $|B|=n$ → $m*n$
  - Ex) #jackets: 5, #pants: 4, the no. ways to put on your clothes is 5*4=20



- **Rule of Sum**
  - #Cases that event A or B occurs
  - If A & B are independent → $m+n$
  - Ex) #jackets: 5, #pants: 4, If one of them is messed at the laundry, it is one of 5+4=9 clothes

1. Basic Counting ⟨2⟩

- **#Elements in Union Sets**
  - $|A \cup B| = |A| + |B| - |A \cap B|$
  - Double counting problem exists!
  - It is a slippery aspect of combinatorics

  ➔ Make it difficult to solve problems via inclusion-exclusion

- **Permutation**
  - An arrangement of n items, where every item appears exactly once
  - $n! = \prod_{i=1}^{n} i = n*(n-1)*(n-2)*\ldots*2*1$
  - ex) How many cases when arranging a,b,c items?
    - abc, acb, bac, bca, cab, cba => 6 cases
  - cf) What if arranging a, b items of length-3 strings under the repetition?
    - ${}_n\Pi_r = n^r$
    - aaa, aab, aba, abb, baa, bab, bba, bbb => 8 cases

2. Recurrence Relation

- Recurrence relations make it easy to count a variety of recursively defined structures

- The recursively defined structures
  - Tree, List, Divide-conquer algorithm, etc.

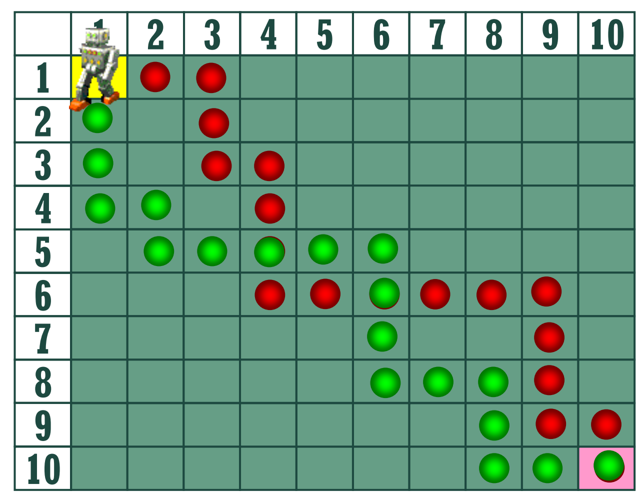
- Recurrence
  - An equation defined in terms of itself
  - Any function can be represented by a recurrence
    - Polynomial function: $a_n = a_{n-1} + 1, a_1=1 \rightarrow a_n = n$
    - Exponential function: $a_n = 2a_{n-1}, a_1=2 \rightarrow a_n = 2^n$
    - Certain weird but interesting function (e.g., Factorial):
      - $a_n = na_{n-1}, a_1=1 \rightarrow a_n = n!$

3. Binomial Coefficient 〈1〉

- The most important class of counting numbers
- $\binom{n}{k}$: #ways to choose k ones out of n things ($_nC_k$)
- Examples

$_nC_k$

  - **Committees** – # of ways to form a k-member committee from n people?
  - **Path Across a Grid** – # of ways to travel from the upper-left corner of an n x m Grid to the lower-right corner walking down and to the right?

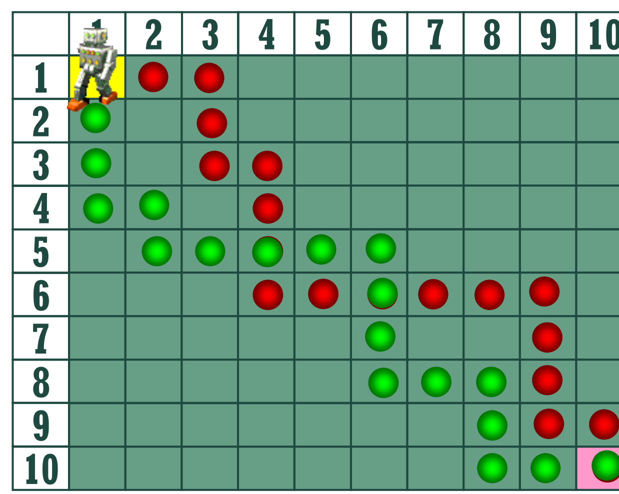


$_{n+m-2}C_{n-1}$
$= {}_{n+m-2}C_{m-1}$

3. Binomial Coefficient 〈1〉

- **Path Across a Grid** – # of ways to travel from the upper-left corner of an n x m Grid to the lower-right corner walking down and to the right?

$${}_{n+m-2}C_{n-1}$$
$$= {}_{n+m-2}C_{m-1}$$



3. Binomial Coefficient 〈2〉

❖ Examples (Cont')

- **Coefficients of $(a+b)^n = (a+b)*(a+b)*\ldots*(a+b)$**
  - What's the coefficient of $a^k b^{n-k}$ term?

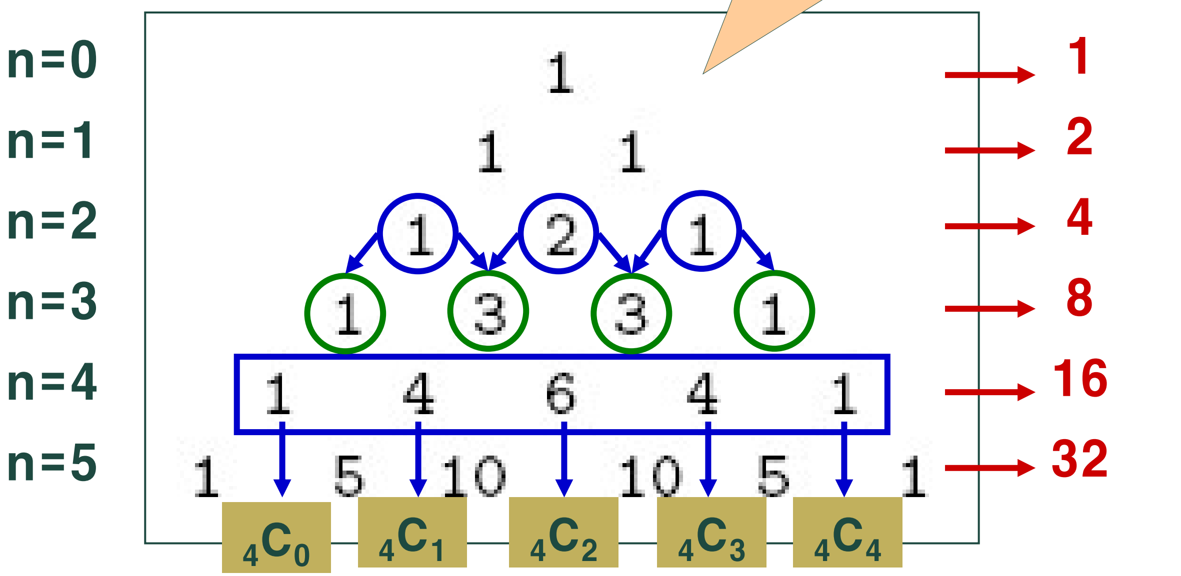
$_nC_k$

- **$(a + b)^3 = 1a^3 + 3a^2b + 3ab^2 + 1b^3$**
  - $(a + b)^3 = (a + b)\ (a + b)\ (a + b)$

- **Pascal' s Triangle**

It represents $_nC_k$!

Sum=2^n

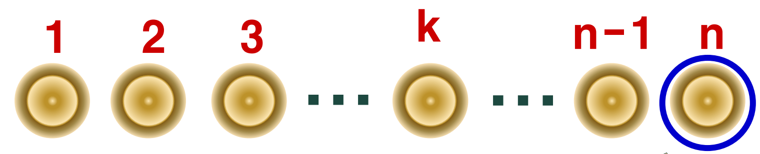


3. Binomial Coefficient 〈3〉

❖ Computing the Binomial Coefficients

$$\begin{bmatrix} n \\ k \end{bmatrix} = \frac{n!}{(n-k)!k!}$$

➔ **But!:** Intermediate calculations (i.e., factorial) can easily cause arithmetic overflow!



❖ Computing by Recurrence Relation

$$\begin{bmatrix} n \\ k \end{bmatrix} = \begin{bmatrix} n-1 \\ k-1 \end{bmatrix} + \begin{bmatrix} n-1 \\ k \end{bmatrix}$$

Consider whether the nth element belongs the chosen k elements!

➔ For the **n**th element, consider two cases:

Case 1: the element belongs to the chosen k elements

Case 2: the element is not in the chosen k elements

3. Binomial Coefficient <4>

❖ Computing by Recurrence Relation – Example

- **A given set {1, 2, 3, 4}; n=4, k = 2, no repetition**

$$\left[\begin{matrix} 4 \\ 2 \end{matrix}\right] = \{ 12, 13, 14, 23, 24, 34 \} = 6$$

- **For the specific element '1'**

① Case 1: '1' belongs to the chosen two elements
➔ Need to choose one element from {2, 3, 4}! $\left[\begin{matrix} 3 \\ 1 \end{matrix}\right]$

② Case 2: '1' does not belong to the chosen set
➔ Need to choose two elements from {2, 3, 4}! $\left[\begin{matrix} 3 \\ 2 \end{matrix}\right]$

- Therefore, #all the possibilities becomes ① + ②

$$\left[\begin{matrix} n \\ k \end{matrix}\right] = \left[\begin{matrix} n-1 \\ k-1 \end{matrix}\right] + \left[\begin{matrix} n-1 \\ k \end{matrix}\right]$$

3. Binomial Coefficient ⟨5⟩

- As to Recurrence relations,
  - The initial condition (term) is essential!
- The best way to evaluate

  ➔ build a Table of all possible values!

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

nCo

nCn

```
long binomial_coefficient(n,m)
{
    int i,j;
    long bc[MAXN][MAXN];

    for (i=0; i<=n; i++) bc[i][0] = 1;

    for (j=0; j<=n; j++) bc[j][j] = 1;


    for (i=1; i<=n; i++)
        for (j=1; j<i; j++)
            bc[i][j] = bc[i-1][j-1] + bc[i-1][j];

    return( bc[n][m] );
}
```

3. Binomial Coefficient <6>

❖ Binomial coefficient table for n=4, m=2

	0	1	2	3	4	
0	1					
1	1	1				
2	1	2	1			…
3	1	3	3	1		
4	1	4	6	4	1	
			…			

bc[2][1] = bc[1][0]+bc[1][1]
bc[3][1] = bc[2][0]+bc[2][1]
bc[3][2] = bc[2][1]+bc[2][2]
bc[4][1] = bc[3][0]+bc[3][1]
bc[4][2] = bc[3][1]+bc[3][2]
bc[4][3] = bc[3][2]+bc[3][3]

❖ main func.

```
int main(void)
{
    int a, b;

    while (1) {
        scanf("%d %d",&a,&b);
        printf("%d₩n",binomial_coefficient(a,b));
    }
    return 0;
}
```

Recursion (and Induction)

- Math. Induction provides a tool to solve Recurrences
- Math. Induction is implemented by Recursion
- Thus, Recursion is the way of solving Recurrences

$$T_n = 2T_{n-1} + 1,\ T_0=0$$

n	0	1	2	3	4	5	6	7	…
T_n	0	1	3	7	15	31	63	127	…

As n increase, T_n increases roughly double. Assume $T_n=2^n-1$!

Solution by Math. Induction!

- Check the validity at n=0: $T_0 = 2^0 - 1 = 0$
- Assume the validity for T_n: $T_n = 2^n - 1$
- After that, check the validity for T_{n+1}:
  $T_{n+1} = 2T_n + 1 = 2(2^n - 1) + 1 = 2^{n+1} - 1$

5. Example – Fibonacci Numbers <1>

Fibonacci numbers are defined by

- $F_0 = 0$
- $F_1 = 1$
- $F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, …

Pseudo-code

```
Fibonacci (n)
if (n=0) then     return (0)
else
            if (n=1)
                        then return (1)
            else
                        return (Fibonacci(n-1) + Fibonacci(n-2))
```

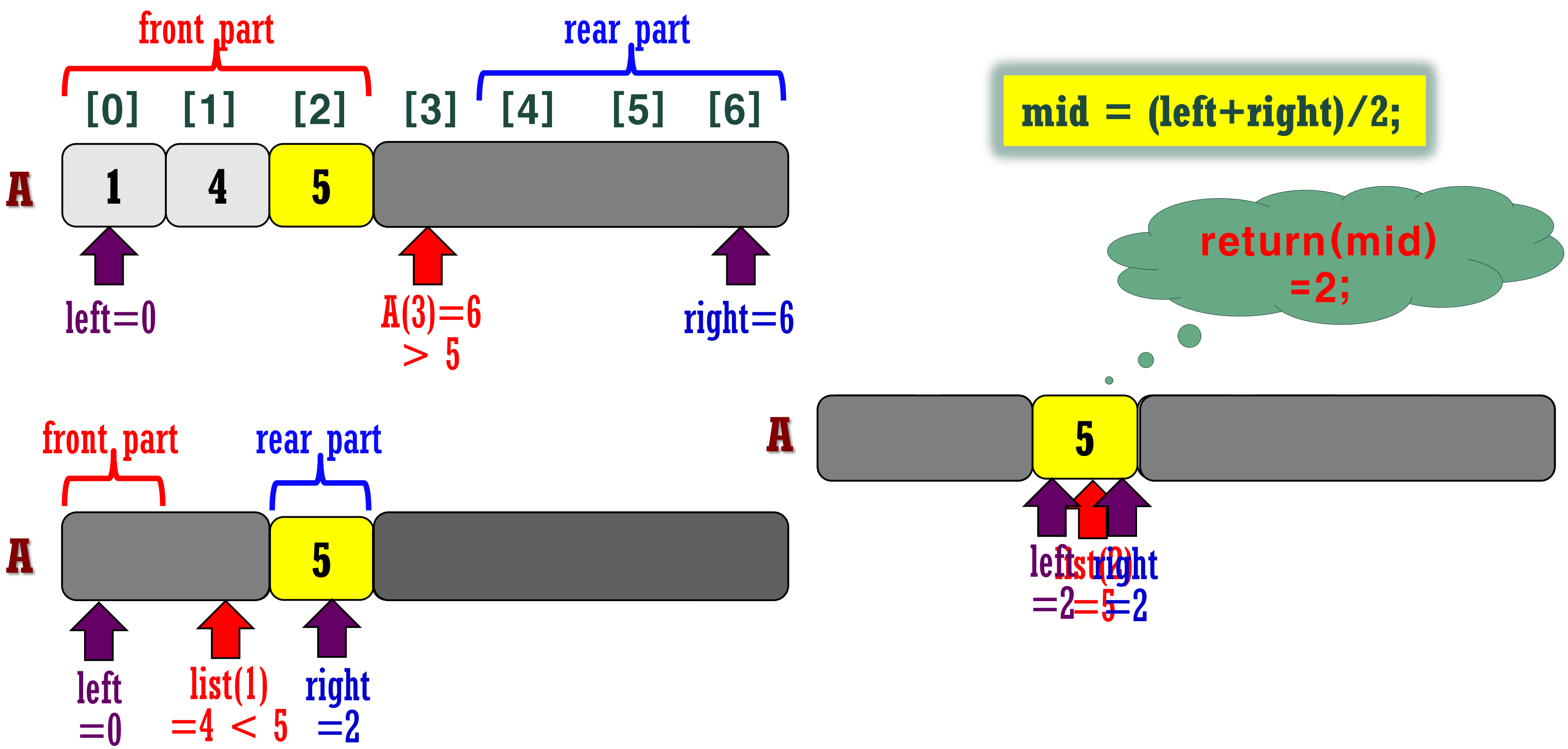
5. Example – Fibonacci Numbers <2>

C Source Code

```
int main(void)
{
        int input_num=0;
        int i;

        while (1){

            printf("Enter the number: ");
            scanf("%d", &input_num);

            if (input_num == 0)
                break;

            printf("result by recursion: ");

            for(i = 0; i <= input_num; i++)
                printf( "%d ", fibonacci_recursion( i ) );

            printf("₩n₩n");

        }

        return 0;
}


int fibonacci_recursion(int num)
{
        if(num < 2)    //f(0)=0, f(1)=1
                return num;
        return fibonacci_recursion(num - 1) + fibonacci_recursion(num - 2); //        f(n) = f(n-1) + f(n-2)

}
```

6. Example – Binary Search <1>

- We want to find an integer X out of n integers.
  - n integers are sorted in ascending order
  - We use Binary search in terms of Recursion.
  - Consider initial condition and recursive structure!



6. Example – Binary Search <2>

❖ Think about Recursive Structure!

- B_SEARCH(A, x, 0, n-1); left=0, right=n-1, mid = (n-1)/2
- If A[mid] > x, call B_SEARCH(A, x, 0, mid-1)
- If A[mid] < x, call B_SEARCH(A, x, mid+1, n-1)
- If A[mid] == x, return(mid)

```
int  B_SEARCH (A[ ], x, left, right) {
     int  mid;
     If  (left <= right)  {
         mid = (left + right) / 2;
         if (x < A[mid])
                B_SEARCH (A[ ], x, left, mid-1);
         else if (x > A[mid])
                B_SEARCH (A[ ], X, mid+1, right);
         else
                return (mid);
     }   }
```