가톨릭대학교 THE CATHOLIC UNIVERSITY OF KOREA Computer SCIENCE

문제해결기법 - 201921725 안성현

하세 다이어그램

<2021/09/04>

문제 소개 및 접근법

1-1] 하세 다이어그램 문제

① 문제

> 집합 X의 부분집합들 간의 포함관계를 나타내는 하세 다이어그램을 자동으로 그리는 패키지를 설계하고 있다. 이 목적을 달성하기 위해서는 k개의 원소를 가진 부분집합, 즉 k-부분집합을 모두 나열하고 각각의 부분집합 A에 대하여, A != X일 경우에 A에 포함되지 않은 원소를 하나 추가하여 만들 수 있는 부분집합을 모두 찾아내는 프로그램이 필요하다. 원소를 n개 가지는 집합 X={1,2,…,n}이고 n<=99일 때 이 프로그램을 구현하라.
>
> (출력은 '사전식 순서'이며 제한 시간은 1.0초이다. 부분집합의 총 개수를 나타내는 C(n,k)+(n-k)C(n,k)는 10^4을 넘지 않는다.)

② 접근법 (소스 간략 설명)

▶ 조합의 '반복적 버전'으로 문제를 풀었다. 사전식 순서로 조합을 출력할 때 나타나는 규칙을 이용하면 된다. 예를 들어 (n,k)가 (6,4)일 때, 출력은 (1|2|3|4 -> 1|2|3|5 -> 1|2|3|6 -> 1|2|4|5->…->3|4|5|6)이 된다. 출력을 자세히 살펴보면 배열의 각 원소마다 최대치가 존재한다. 마지막 원소의 최대치는 6이다. 마지막에서 두 번째 원소의 최대치는 5이다. 같은 이치로 첫 번째 원소의 최대치는 3이다. 출력 배열을 kset으로 정의하고 인덱스는 1부터 이용한다고 할 때, 위 규칙을 일반화하면 kset[i]=n-k+1일 때 최대치가 된다. 예를 들어 kset[1]=3일 때 최대치가 되는데, 첫 번째 원소는 3보다 클 수 없기 때문이다. 이제 kset[i]가 최대치가 될 때까지 kset[i]를 1씩 올리며 출력을 하면 된다. i의 초기값은 k이다. 만약 최대치가 되면 인덱스 i이하의 원소들을 업데이트한다. (1|2|3|6)을 (1|2|4|5)로 만들겠다는 의미이다. 이것이 가능하려면 i를 k부터 kset[i]가 최대치가 아닐 때까지 낮추고 kset[i]에 1을 더한다. 그리고 kset[i]~kset[k]까지는 공차가 1인 등차수열이라고 생각하면 된다. 예를 들어 (1|2|3|6), i=4라면 i는 3까지 낮춰지고 kset[3]=4, kset[4]=5가 되어 (1|2|4|5)가 된다. 이 **작업**을 반복해서 (3|4|5|6), i=1이 되면 모든 출력을 마치게 된다. k+1부분집합은 출력한 k부분집합을 인덱스로 가지는 out배열의 원소에 1을 넣고 나머지 원소에 1을 추가해가며 출력하면 된다.

소스 코드와 실행 결과

2-1] 소스 코드와 실행 결과

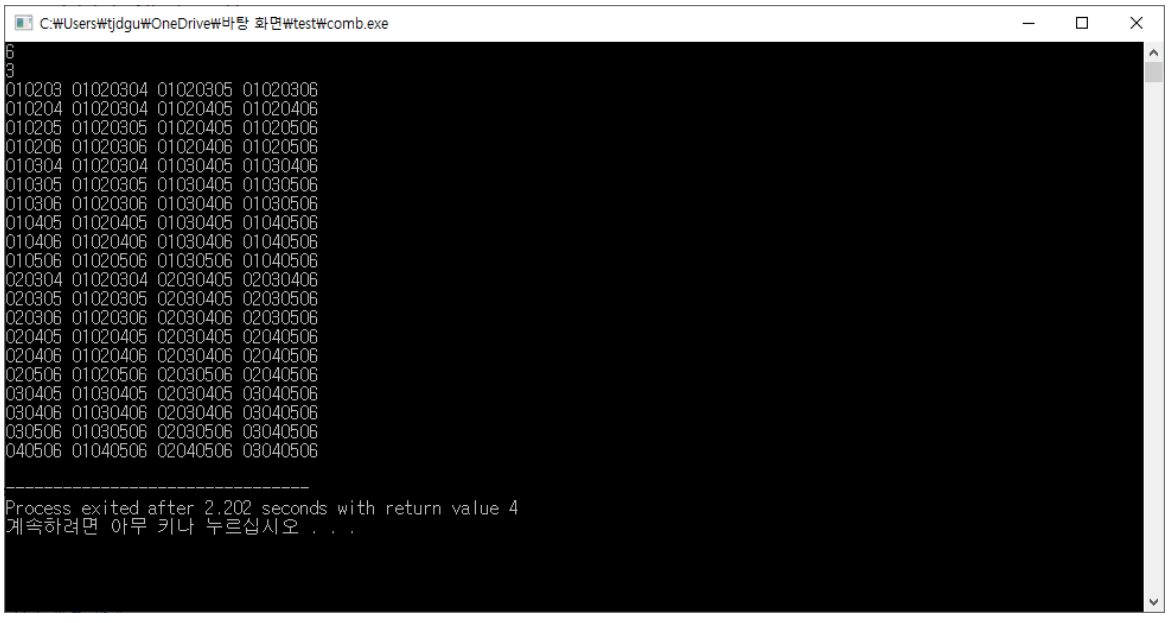
① 코드

```
#include <stdio.h>
#include <stdlib.h>

// k+1-부분집합 출력
void print_kplus1(int * kset, int n, int k) {
	int* out = (int*)malloc(sizeof(int)*n + 1); // 1~n 중 출력된 것은 1처리
	int num = 0; // for-if문이 동작한 횟수

	// 출력된 것은 1처리
	int t;
	for (t = 1; t <= k; t++) {
		out[kset[t]] = 1;
	}

	int i;
	for (i = 1; i <= n; i++) {
		// 출력 안 된 인덱스와 함께 출력 시킴
		if (out[i] != (num + 1)) {
			out[i] = ++num; // 출력 안 된 인덱스도 num+1처리

			// k+1-부분집합 원소 출력
			for (t = 1; t <= n; t++) {
				if (out[t] == num) {
					printf("%02d", t);
				}
			}
			// num에 따라 띄어쓰기 여부가 결정됨 (마지막 출력은 적용x)
			if (num != n - k) {
				printf(" ");
			}

			// 원상복귀 (처음 출력된 인덱스는 num+1 처리)
			for (t = 1; t <= k; t++) {
				out[kset[t]] = num + 1;
			}
		}
	}
}

// 한 줄씩 부분집합 원소 출력
int subset(int n, int k) {
	// 아무 것도 뽑지 않음 -> 공집합 -> 출력 후 빠져 나옴
	if (k == 0) {
```

```
                printf("00 ");
                // k+1-부분집합 출력
                int* kplus1 = (int*)malloc(sizeof(int) * 1);
                int i;
                for (i = 1; i <= n; i++) {
                        if (i < n)
                                printf("%02d ", i);
                        else
                                printf("%02d", i);
                }
                printf("₩n");
                return 0;
        }

        // 출력할 부분집합이 담길 배열
        int* kset = (int*)malloc(sizeof(int)*(k + 1));
        int i;

        // 첫 번째 출력할 부분집합 생성
        // kset[1]=1, kset[2]=2, ... kset[k]=k 로 매치
        // ex) (n,k)=(6,4) => 1|2|3|4 배열이 생성
        for (i = 1; i <= k; i++)
                kset[i] = i;

        int finish = 0; // 부분집합 출력이 끝나면 finish->1
        while (!finish) {
                // k부분집합 출력
                for (i = 1; i <= k; i++) {
                        printf("%02d", kset[i]);
                }
                printf(" ");
                // k+1-부분집합 출력
                print_kplus1(kset, n, k);
                printf("₩n");

                // kset[idx]가 최대치인지 검사
                // 최대치 -> kset[--idx]가 최대치인지 검사
                // ex) idx=4, 1|2|3|'6' 최대치-> idx=3, 1|2|'3'|6 최대치 아님
                int idx;
                for (idx = k; kset[idx] == n - k + idx; --idx) {
                        // 첫 번째 원소도 최대치임 -> 모두 출력됨
                        // ex)idx=1, 3|4|5|6 <사전식 출력 완료>
                        if (idx == 1) {
                                finish = 1;
                                break;
                        }
                }
                // 출력이 끝나지 않았으면 진행
                if (!finish) {
                        // kset[idx]를 1높여서 업데이트 ex) idx=3, 1|2|'3'|6 -> 1|2|'4'|6
                        kset[idx]++;
                        // idx를 기준으로 뒤의 원소들도 업데이트 ex) idx=3, 1|2|'4'|6
<'4' 기준으로 뒤의 원소들 1씩 추가> -> 1|2|4|'5'
```

```
                        for (i = idx + 1; i <= k; i++)
                                kset[i] = kset[i - 1] + 1;
                }
        }
}


int main()
{
        int n, k;

        // n 입력 받기
        scanf("%d", &n);

        // k 입력 받기
        scanf("%d", &k);

        // 부분집합 출력 문제 풀기
        subset(n, k);

        return 0;
}
```

② DEV-C++ 컴파일러 이용 실행 결과



입력: 집합의 크기(6), 부분집합의 원소 개수(3)

출력: C(6.3)+ (6-3) x C(6,3) = 80개의 부분집합