민 선 생 코 딩 학 원 훈 련 반

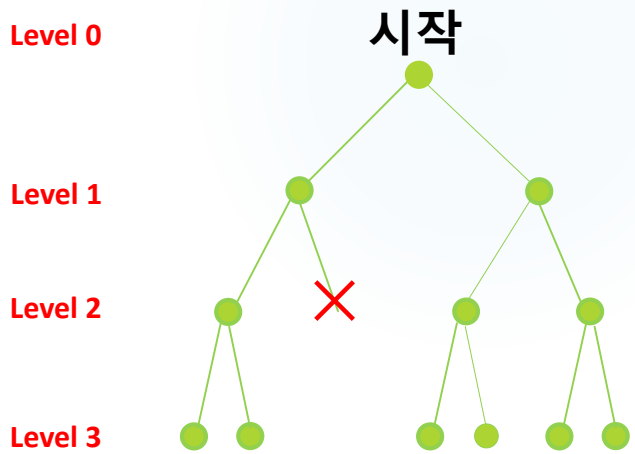
수업노트 LV-23

배우는 내용

1. **가지치기**
2. **via로 중복선택 막기**

가지치기란?

- return 조건을 추가하는 것
- 진입하고 싶지 않는 곳이 있을 때, 인위적으로 막는다.



가지치기 방법1 – 진입 후 바로 나가기

- 다량의 A B C 세 종류의 카드를 가지고 있다.
  이 중 3장을 뽑을 때 모든 조합을 출력하라
  (단, B로 시작하는 조합 제외)

```cpp
#include <iostream>
using namespace std;

char path[10];

void run(int level)
{
    if (path[0] == 'B') return;

    if (level == 3) {
        cout << path << endl;
        return;
    }

    for (int i = 0; i < 3; i++) {
        path[level] = 'A' + i;
        run(level + 1);
        path[level] = 0;
    }
}

int main()
{
    run(0);
    return 0;
}
```

```
AAA
AAB
AAC
ABA
ABB
ABC
ACA
ACB
ACC
CAA
CAB
CAC
CBA
CBB
CBC
CCA
CCB
CCC
```

가지치기 방법2 – 아예 진입안하기

- 다량의 A B C 세 종류의 카드를 가지고 있다.
  이 중 3장을 뽑을 때 모든 조합을 출력하라
  (단, B로 시작하는 조합 제외)

```
#include <iostream>
using namespace std;

char path[10];

void run(int level)
{
    if (level == 3) {
        cout << path << endl;
        return;
    }

    for (int i = 0; i < 3; i++) {
        //!는 반대를 뜻한다.
        if (!(level == 0 && 'A' + i == 'B')) {
            path[level] = 'A' + i;
            run(level + 1);
            path[level] = 0;
        }
    }
}

int main()
{
    run(0);
    return 0;
}
```

가지치기의 의미

- 진입할 필요가 없는 곳에는 if로 막아준다.
- 가지치기를 잘 할 수록 프로그램 성능이 좋아진다.
  (쓸대없는 진입을 막아주기 때문)
- 가지치기는
  정확한 조건의 if문을, 정확한 위치에 적어주는 것이 중요

가지치기 예제 – 진입 후 바로 나가기

- 다량의 A B C 세 종류의 카드를 가지고 있다. 이 중 3장을 뽑을 때 모든 조합을 출력하라 **(단, 같은 카드를 연속 2장 뽑으면 안됨)**

```cpp
#include <iostream>
using namespace std;

char path[10];

void run(int level)
{
    //현재값 : path[level - 1]
    //이전값 : path[level - 2]
    //현재값과 이전값이 같으면 return
    if (level >= 2 && path[level - 2] == path[level - 1]) return;

    if (level == 3) {
        cout << path << endl;
    }

    for (int i = 0; i < 3; i++) {
        path[level] = 'A' + i;
        run(level + 1);
        path[level] = 0;
    }
}

int main()
{
    run(0);
    return 0;
}
```

```
ABA
ABC
ACA
ACB
BAB
BAC
BCA
BCB
CAB
CAC
CBA
CBC
```

가지치기 예제 - 아예 진입안하기

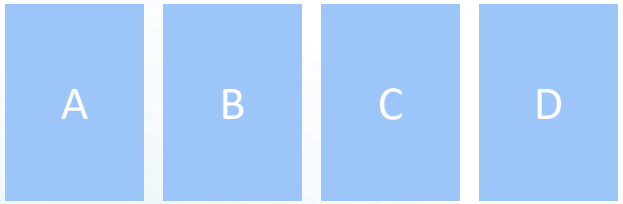
- 왼쪽방법은 진입 후 return하는 방법
  오른쪽 방법은 진입조차 하지 않는 방법
- 결과는 같으며, 둘 다 좋은 방법이다.
  두개의 명확히 차이를 알아야 한다

```cpp
#include <iostream>
using namespace std;

char path[10];

void run(int level)
{
    if (level >= 2 && path[level - 2] == path[level - 1]) return;

    if (level == 3) {
        cout << path << endl;
        return;
    }

    for (int i = 0; i < 3; i++) {
        path[level] = 'A' + i;
        run(level + 1);
        path[level] = 0;
    }
}

int main()
{
    run(0);
    return 0;
}
```

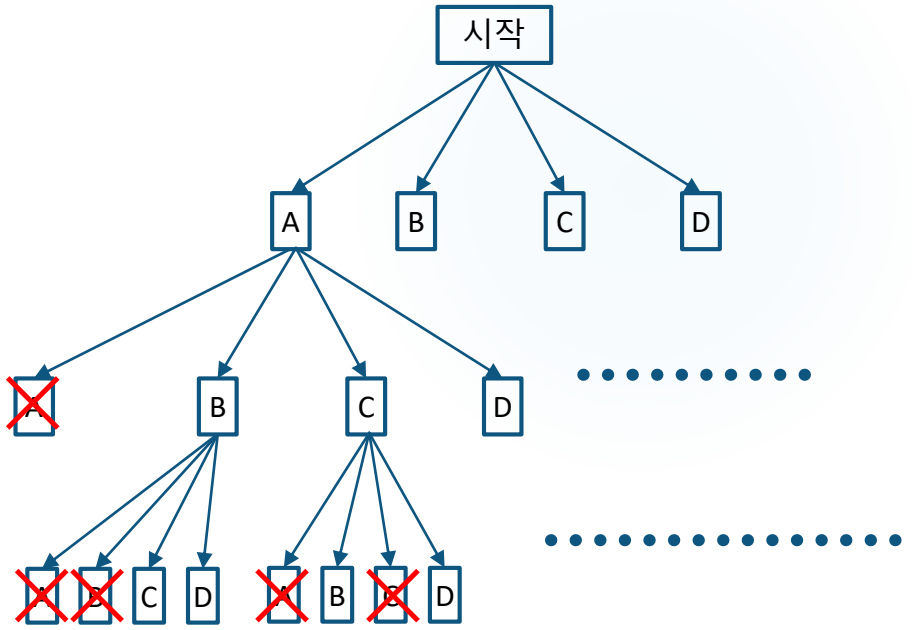
```cpp
#include <iostream>
using namespace std;

char path[10];

void run(int level)
{
    if (level == 3) {
        cout << path << endl;
        return;
    }

    for (int i = 0; i < 3; i++) {
        //path[level - 1] : 현재값
        //'A' + i : 다음값
        if (path[level - 1] != 'A' + i) {
            path[level] = 'A' + i;
            run(level + 1);
            path[level] = 0;
        }
    }
}

int main()
{
    run(0);
    return 0;
}
```

[중요] 네 장 중 한번씩만 세장 뽑기

- 네 장의 카드 중
  한번 뽑았던 카드는 다시 안뽑고, 세장 뽑는 방법



정답
ABC
ABD
ACB
ACD
ADB
…
DCB



[중요] via 전역배열을 이용하여 중복선택 막기

- 세장의 카드 중
  한번 뽑았던 카드는 다시 안뽑는 방법

- DFS를 쓸 때 자주 사용하는 방법

```cpp
#include <iostream>
using namespace std;

char path[10];
int via[10];

void run(int level)
{
    if (level == 3) {
        cout << path << endl;
        return;
    }

    for (int i = 0; i < 4; i++) {
        //i를 처음 선택한다면
        if (via[i] == 0) {
            via[i] = 1;
            path[level] = 'A' + i;
            run(level + 1);
            path[level] = 0;
            via[i] = 0;
        }
    }
}

int main()
{
    run(0);
    return 0;
}
```

```
ABC
ABD
ACB
ACD
ADB
ADC
BAC
BAD
BCA
BCD
BDA
BDC
CAB
CAD
CBA
CBD
CDA
CDB
DAB
DAC
DBA
DBC
DCA
DCB
```