알고리즘 과제 4

m-coloring problem

제출 마감일 : 2016.12.12

담당교수 : 최지웅 교수님

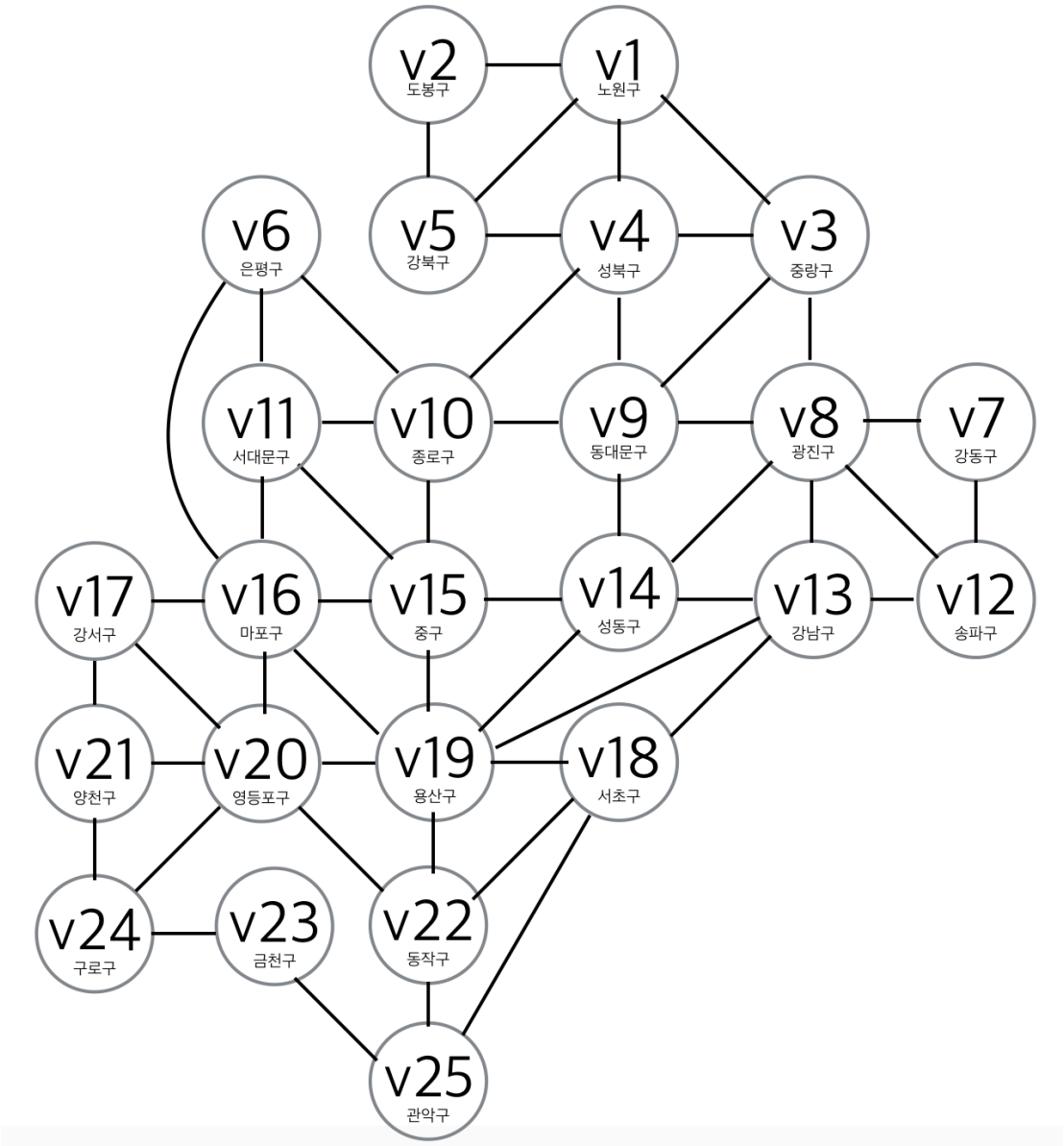
소속학부 : 컴퓨터학부

학번 : 20142577

이름 : 홍상원

1. 과제 개요
- 서울지도를 기반으로 m-coloring 알고리즘을 실행시켜본다.
1) 최소 m 을 구한다. (m: 색의 개수)
2) 최소 m 일 때, 해가 몇 개인지 구한다.
- m-coloring problem : m 개의 색을 사용하여 인접한 지역이 같은 색이 되지 않도록, 지도를 색칠하는 방법이 얼마나 있는지 구하는 문제
- 서울지도.PNG

2) 그래프 구현
- 서울지도.PNG 를 그래프로 나타낸 것이다.



- 정점: 구
- 이음선: 인접한 구들을 연결
- m-coloring 문제는 색을 통해 지역을 구분시키는 것이 목적이다. 점으로 인접하는 지역 간에는 같은 색을 칠하더라도 지역 간에 구분이 된다. 따라서 점으로 접하는 지역 간에는 인접하지 않았다고 간주하여 이음선으로 연결하지 않았다.
ex) 중구 & 동대문구, 동작구 & 구로구

3) 구현내용

- 문제: 비방향그래프에서 m 개의 색만 사용하여 인접한 정점이 같은 색이 되지 않게 정점을 색칠할 수 있는 모든 방법을 구하라.
- 입력: 양의 정수 n 과 m, 그리고 n 개의 정점을 가진 비방향그래프. 그래프는 행과 열의 인덱스가 모두 1 부터 n 까지인 2 차원 배열 W 로 표현된다. 여기서 i 번째 정점과 j 번째 정점 사이에 이음선이 존재한다면 W[i][j]는 true 이고, 그렇지 않다면 false 이다.
- 출력: 최대 m 개의 색을 가지고 인접한 정점이 같은 색이 되지 않게 그래프에 색칠하는 가능한 모든 경우. 출력은 인덱스가 1 부터 n 까지인 배열 vcolor 이다. 여기서 vcolor[i] 값은 i 번째 정점에 할당된 색(1 부터 m 사이의 정수)이다.
- 소스코드

[m-coloring.cpp]

```
#include <iostream>
#define N 25 //서울지도 구 개수 == 정점의 개수
using namespace std;

//유망한지 여부를 검사
bool promising(int i, bool W[][N + 1], int vcolor[]) {
	int j = 1;
	bool sw = true;
	//인접한 정점에 이 색이 이미 칠해져 있는지 점검한다.
	while(j < i && sw) {
		if(W[i][j] && vcolor[i] == vcolor[j])
			sw = false;
		j++;
	}

	return sw;
}

//m-coloring 을 진행
int m_coloring(int i, int m, bool W[][N + 1], int vcolor[]) {
	int color;
	static int count = 0;//해의 개수 구하기
	if(promising(i, W, vcolor)) {
		if(i == N) {//모든 정점이 할당되었을 때
			/*
			for(int k = 1; k <= N; k++)
				cout << vcolor[k] << " ";
			cout << endl;
			*/
			count++;//해의 개수 세기
```

```
			}
			else {	//다음 정점에 모든 색을 시도해본다.
				for(color = 1; color <= m; color++) {
					vcolor[i + 1] = color;
					m_coloring(i + 1, m, W, vcolor);
				}
			}
		}

		return count;
}

int main(void) {
		bool W[N + 1][N + 1] = {//서울지도로 나타낸 그래프를 인접행렬로 저장
		{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
		{0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},//1
		{0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},//2
		{0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},//3
		{0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},//4
		{0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},//5
		{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0},//6
		{0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},//7
		{0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},//8
		{0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},//9
		{0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},//10
		{0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0},//11
		{0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},//12
		{0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0},//13
		{0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0},//14
		{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0},//15
		{0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0},//16
		{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0},//17
		{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1},//18
		{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0},//19
		{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0},//20
		{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0},//21
		{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1},//22
		{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1},//23
		{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0},//24
		{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0}//25
		};
```

```
        int m = 4;//최소 m 의 개수 = 4
        int vcolor[N + 1] = {0};
        int  count = m_coloring(0, m, W, vcolor);
        cout << "해의 개수: " << count << endl;
        return 0;
}
```

4) 최소 m(색의 개수) 구하기

- m = 3 일 때

```
hongsang-won-ui-MacBook-Pro:ch05 Frodo$ g++ m_coloring.cpp
hongsang-won-ui-MacBook-Pro:ch05 Frodo$ ./a.out
해의  개수 : 0
```

-> 3 개 이하의 색으로는 인접한 영역을 다른 색깔로 칠하면서 서울지도를 모두 색칠할 수 없다.

- m = 4 일 때

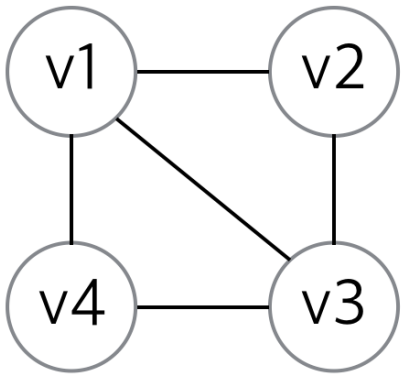
```
hongsang-won-ui-MacBook-Pro:ch05 Frodo$ g++ m_coloring.cpp
hongsang-won-ui-MacBook-Pro:ch05 Frodo$ ./a.out
해의  개수 : 30766080
```

-> 4 개의 색부터는 인접한 영역을 다른 색깔로 칠하면서 서울지도를 모두 색칠할 수 있다.

- 따라서 최소 m 은 4 이고 이때 해의 개수는 30766080 개 이다.

5) 실행결과

- 올바르게 작동하는 프로그램인지 간단한 그래프를 입력하여 실행결과를 통해 확인해 본다.



- 위 그래프를 입력하는 수정된 소스코드

[m_coloring_test.cpp]

```
#include <iostream>
#define N 4 //수정
using namespace std;
```

```
bool promising(int i, bool W[][N + 1], int vcolor[]) {
	int j = 1;
	bool sw = true;
	while(j < i && sw) {
		if(W[i][j] && vcolor[i] == vcolor[j])
			sw = false;
		j++;
	}

	return sw;
}

int m_coloring(int i, int m, bool W[][N + 1], int vcolor[]) {
	int color;
	static int count = 0;
	if(promising(i, W, vcolor)) {
		if(i == N) {
			for(int k = 1; k <= N; k++)
				cout << vcolor[k] << " ";
			cout << endl;
			count++;
		}
		else {
			for(color = 1; color <= m; color++) {
				vcolor[i + 1] = color;
				m_coloring(i + 1, m, W, vcolor);
			}
		}
	}

	return count;
}

int main(void) {
	bool W[N + 1][N + 1] = {
		{0, 0, 0, 0, 0},
		{0, 0, 1, 1, 1},
		{0, 1, 0, 1, 0},
		{0, 1, 1, 0, 1},
		{0, 1, 0, 1, 0}
	};//수정
	int m = 3;
```

```
        int vcolor[N + 1] = {0};
        int  count = m_coloring(0, m, W, vcolor);
        cout << "해의 개수: " << count << endl;
        return 0;
}
```

- 출력결과