Dynamic Programming 1

나정휘

https://justiceHui.github.io/

목차

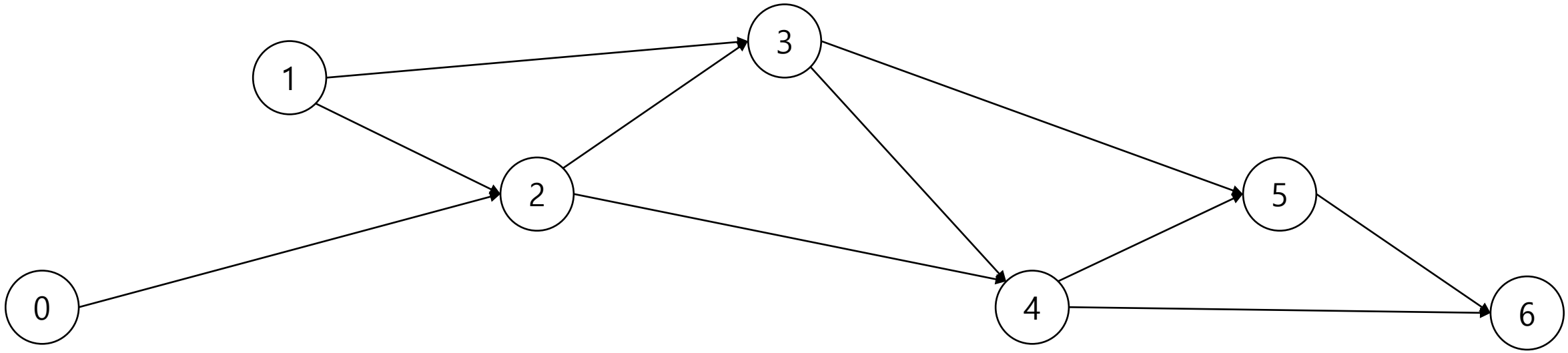
DAG에서의 DP

DAG에서의 DP

- DAG와 DP의 관계
  - DAG: 사이클이 없는 방향 그래프
    - 사이클이 없음
    - 위상 정렬을 할 수 있음
  - DP: 작은 문제의 답을 이용해 큰 문제의 답을 구하는 방법
    - 작은 문제와 큰 문제 간의 연결 관계 → DAG
    - 부분 문제를 위상 정렬 순서대로 해결
      - 재귀 함수 + 메모이제이션

동적 계획법

- DAG와 DP의 관계
  - ex. 피보나치 수
    - 위상 정렬 상에서 앞선 문제의 답을 구해야 뒤에 있는 문제의 답을 구할 수 있음
    - 사이클 없음



동적 계획법

- DAG와 DP의 관계
  - 일반적인 그래프에서 풀지 못하는 문제를 DAG에서는 DP를 이용해 빠르게 해결할 수 있음
    - 최장 경로, S-T 경로의 개수
      - 일반적으로 다항 시간에 해결할 수 없지만 DAG에서는 O(|V| + |E|)
  - 몇몇 DP 문제는 DAG로 생각해서 푸는 것이 편한 경우도 있음
    - 최장 공통 부분 문자열: DAG에서의 최장 경로
    - 동전 교환 경우의 수: DAG에서의 경로 개수
    - ...

DAG에서의 DP

- BOJ 1005 ACM Craft
  - DAG가 주어지면 정점 w까지 가는 가장 긴 경로의 길이를 구하는 문제
  - 단, 간선이 아닌 정점에 가중치가 붙어 있음

  - 점화식
    - D[x] = x까지 가는 최장 경로의 길이
    - u에서 v로 가는 간선이 있다면 D[v] ← D[u] + A[v]

  - 위상 정렬 순서대로 점화식을 계산하면 됨

DAG에서의 DP

```cpp
#include <bits/stdc++.h>
using namespace std;

int N, M, A[1010], C[1010], D[1010];
vector<int> G[1010];

void Solve(){
    cin >> N >> M;
    for(int i=1; i<=N; i++) cin >> A[i];
    for(int i=1,s,e; i<=M; i++) cin >> s >> e, G[s].push_back(e), C[e]++;

    queue<int> Q;
    for(int i=1; i<=N; i++) D[i] = A[i];
    for(int i=1; i<=N; i++) if(!C[i]) Q.push(i);
    while(!Q.empty()){
        int v = Q.front(); Q.pop();
        for(auto i : G[v]){
            D[i] = max(D[i], D[v] + A[i]);
            if(!--C[i]) Q.push(i);
        }
    }
    int T; cin >> T;
    cout << D[T] << "\n";
}
```

DAG에서의 DP

- 연습 문제
  - BOJ 24888 노트 조각
  - BOJ 16297 Eating Everything Efficiently

질문?

트리에서의 DP

트리에서의 DP

- Tree DP
  - Rooted Tree는 DAG
  - 일반적인 구조
    - “D[v] = v를 루트로 하는 서브 트리의 정답”으로 정의
    - 자식 정점들의 DP값을 잘 합치는 방법을 찾아야 함

트리에서의 DP

- BOJ 15681 트리와 쿼리
  - 루트가 있는 트리가 주어지면, 각 정점을 루트로 하는 서브 트리의 정점 개수를 구하는 문제
    - 각 정점의 (자손 정점 개수) + 1를 구하는 문제

  - D[v] = v를 루트로 하는 서브 트리의 크기
    - v의 자식 정점 c_1, c_2, … , c_k가 있다고 하면
    - D[v] = D[c_1] + D[c_2] + … + D[c_k] + 1
    - 리프 정점부터 크기를 구하면 됨

트리에서의 DP

```cpp
#include <bits/stdc++.h>
using namespace std;

int N, R, Q, S[101010];
vector<int> G[101010];

void DFS(int v, int b=-1){
    S[v] = 1;
    for(auto i : G[v]) if(i != b) DFS(i, v), S[v] += S[i];
}

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N >> R >> Q;
    for(int i=1,u,v; i<N; i++) cin >> u >> v, G[u].push_back(v), G[v].push_back(u);
    DFS(R);
    for(int i=1,t; i<=Q; i++) cin >> t, cout << S[t] << "\n";
}
```

트리에서의 DP

- BOJ 1949 우수 마을
  - 정점마다 가중치가 있는 트리에서 정점을 몇 개 선택해야 함
  - 이때 인접한 두 정점을 모두 선택할 수는 없음 (독립 집합)
  - 선택한 정점들의 가중치의 합을 최대화

  - 선형일 때의 문제를 먼저 풀어보자.
    - 배열 A[]에서 인접한 두 원소를 모두 선택하지 않으면서 선택한 원소의 합을 최대화

  - A[1..i-1]만 고려했을 때의 정답을 알고 있을 때, A[i]를 추가했을 때의 정답을 구해야 함

트리에서의 DP

- BOJ 1949 우수 마을
  - 선형일 때의 문제를 먼저 풀어보자.
    - 배열 A[]에서 인접한 두 원소를 모두 선택하지 않으면서 선택한 원소의 합을 최대화

  - 점화식
    - A[1..i-1]만 고려했을 때의 정답을 알고 있을 때, A[i]를 추가했을 때의 정답을 구해야 함
    - 즉, 점화식의 상태에서 마지막 원소의 선택 여부도 함께 저장해야 함

    - D[i][0] = A[1..i]만 고려했을 때 A[i]를 포함하지 않는 상황에서의 최대 점수
    - D[i][1] = A[1..i]만 고려했을 때 A[i]를 포함하는 상황에서의 최대 점수

    - D[i][0] = max(D[i-1][0], D[i-1][1])
    - D[i][1] = D[i-1][0] + A[i]

트리에서의 DP

- BOJ 1949 우수 마을
  - 트리에서도 동일한 방법으로 해결 가능

  - 점화식
    - D[v][0] = v를 루트로 하는 서브 트리에서 v를 선택하지 않았을 때의 최대 점수
    - D[v][1] = v를 루트로 하는 서브 트리에서 v를 선택했을 때의 최대 점수

    - D[v][0] = max(D[c_1][0], D[c_1][1]) + max(D[c_2][0], D[c_2][1]) + max(D[c_3][0], C[c_3][1]) + ...
    - D[v][1] = A[v] + D[c_1][0] + D[c_2][0] + D[c_3][0] + ...

트리에서의 DP

```cpp
#include <bits/stdc++.h>
using namespace std;

int N, A[10101], D[10101][2];
vector<int> G[10101];

void DFS(int v, int b=-1){
    D[v][1] += A[v];
    for(auto i : G[v]){
        if(i == b) continue;
        DFS(i, v);
        D[v][0] += max(D[i][0], D[i][1]);
        D[v][1] += D[i][0];
    }
}

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N;
    for(int i=1; i<=N; i++) cin >> A[i];
    for(int i=1,u,v; i<N; i++) cin >> u >> v, G[u].push_back(v), G[v].push_back(u);
    DFS(1);
    cout << max(D[1][0], D[1][1]);
}
```

트리에서의 DP

- 연습 문제
  - BOJ 2213 트리의 독립집합
  - BOJ 23089 사탕나무

질문?

구간에 대한 DP

구간에 대한 DP

- BOJ 11049 행렬 곱셈 순서
  - N개의 행렬이 주어짐
  - 행렬의 순서를 바꾸지 않고 N개의 행렬을 모두 곱하는데 필요한 곱셈의 최소 횟수
  - A*B 행렬과 B*C 행렬을 곱하면 A*B*C번의 곱셈이 필요하고, 그 결과는 A*C 크기의 행렬

  - 예시
    - 3개의 행렬 (5, 3), (3, 2), (2, 6)이 주어졌을 때
    - 앞에 있는 2개를 먼저 곱하면 5*3*2 + 5*2*6 = 90번
    - 뒤에 있는 2개를 먼저 곱하면 3*2*6 + 5*3*6 = 126번

구간에 대한 DP

- BOJ 11049 행렬 곱셈 순서
  - 이 문제는 최적 부분 구조가 성립함
    - 첫 번째부터 N번째 행렬을 최소 비용으로 곱하는 것은
    - 적당한 i에 대해, 1~i번째 행렬을 곱한 행렬과 i+1~N번째 행렬을 곱한 행렬을 곱하는 것과 동일
    - 따라서 D[1][N] = min{ D[1][i] + D[i+1][N] + 곱셈 횟수 }

  - D[i][j] = i번째 행렬부터 j번째 행렬을 곱하는데 필요한 곱셈 횟수
    - D[i][j] = min{ D[i][k] + D[k+1][j] + R[i]*C[k]*C[j] }
    - D[i][j]를 계산하기 위해 필요한 부분 문제는 항상 곱해야 하는 행렬이 더 적음
    - 따라서 j – i가 작은 부분 문제부터 계산

구간에 대한 DP

- BOJ 11049 행렬 곱셈 순서
  - 시간 복잡도
    - 부분 문제의 개수: $O(N^2)$
    - 각 부분 문제의 답을 구하기 위해 필요한 시간: $O(N)$
    - 따라서 전체 시간 복잡도는 $O(N^3)$

    - $T(N) = \sum_{i=1}^{N} \sum_{j=i}^{N} (j - i) = \sum_{d=0}^{N} d(N - d) = N \sum d - \sum d^2$
    - $= N^2(N + 1)/2 - N(N + 1)(2N + 1)/6$
    - $\lim_{n \to \infty} \frac{T(N)}{N^3} = \frac{1}{2} - \frac{2}{6} = \frac{1}{6}$
    - $\therefore T(N) \in \Theta(N^3)$

구간에 대한 DP

```cpp
#include <bits/stdc++.h>
using namespace std;
using ll = long long;

ll N, R[555], C[555], D[555][555];

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N;
    for(int i=1; i<=N; i++) cin >> R[i] >> C[i];
    memset(D, 0x3f, sizeof D);
    for(int i=1; i<=N; i++) D[i][i] = 0;
    for(int i=2; i<=N; i++) D[i-1][i] = R[i-1] * R[i] * C[i];
    for(int d=2; d<=N; d++){
        for(int i=1; i+d-1<=N; i++){
            int j = i + d - 1;
            for(int k=i; k<j; k++){
                D[i][j] = min(D[i][j], D[i][k] + D[k+1][j] + R[i] * C[k] * C[j]);
            }
        }
    }
    cout << D[1][N];
}
```

연습 문제

- BOJ 11066 파일 합치기
- BOJ 12013 248 게임
- BOJ 2449 전구

질문?

확률/기댓값 DP

확률/기댓값 DP

- BOJ 15488 나이트가 체스판을 벗어나지 않을 확률
  - N*N 크기의 체스판 위에 나이트가 있음
  - 나이트는 체스판 밖으로 나갈 수 있지만, 한 번 나가면 다시 들어올 수 없음
  - 나이트는 매번 이동할 수 있는 8가지 방법 중 균등한 확률로 한 가지를 골라 이동
  - 나이트가 K번 이동한 후에 체스판 위에 있을 확률

  - 점화식
    - D[k][x][y] = k번 점프해서 (x, y)에 도달할 확률
    - 시작 지점이 (i, j)이면 D[0][i][j] = 1, 다른 모든 D[0][*][*] = 0
    - D[k][x][y]/8 → D[k+1][x+dx][y+dy]

확률/기댓값 DP

```cpp
#include <bits/stdc++.h>
using namespace std;
constexpr int di[] = {-2, -2, -1, 1, 2, 2, 1, -1};
constexpr int dj[] = {-1, 1, 2, 2, 1, -1, -2, -2};

int N, X, Y, K;
double D[55][55][55];
bool Bound(int i, int j){ return 1 <= i && i <= N && 1 <= j && j <= N; }

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N >> X >> Y >> K;
    D[0][X][Y] = 1;
    for(int k=1; k<=K; k++){
        for(int i=1; i<=N; i++){
            for(int j=1; j<=N; j++){
                for(int d=0; d<8; d++){
                    int r = i + di[d], c = j + dj[d];
                    if(Bound(r, c)) D[k][r][c] += D[k-1][i][j] / 8;
                }
            }
        }
    }
    double R = 0;
    for(int i=1; i<=N; i++) for(int j=1; j<=N; j++) R += D[K][i][j];
    cout << fixed << setprecision(20) << R;
}
```

연습 문제

- BOJ 1344 축구
- BOJ 20938 반짝반짝

질문?

과제

- 필수
  - 1005 ACM Craft
  - 24888 노트 조각
  - 15681 트리와 쿼리
  - 1949 우수 마을
  - 11049 행렬 곱셈 순서
  - 11066 파일 합치기
  - 15488 나이트가 체스판을 벗어나지...
  - 1344 축구
- 심화
  - 16297 Eating Everything...
  - 2213 트리의 독립집합
  - 34089 사탕나무
  - 12013 248 게임
  - 2449 전구
  - 20938 반짝반짝