2022-1학기 스터디 #5

나정휘

https://justiceHui.github.io/

목차

- 수학적 귀납법
- 분할 정복
- 분할 정복의 예시 (4문제)

수학적 귀납법

수학적 귀납법

- 자연수에 대한 명제 P(n)이 모든 자연수에 대해 성립함을 증명
  - 어떤 정수 n_0에 대해, $n \geq n_0$을 만족하는 모든 n에 대해 P(n)이 성립

- (base case) $P(n_0)$이 성립함을 증명
- (inductive step) P(k)가 성립하면 P(k+1)이 성립함을 증명
  - k는 n_0보다 크거나 같은 정수

- $n \geq n_0$인 모든 정수 n에 대해 P(n)이 성립
  - $P(n_0) \Rightarrow P(n_0+1) \Rightarrow P(n_0+2) \Rightarrow P(n_0+3) \Rightarrow \cdots$

수학적 귀납법

- $1 + 3 + 5 + \cdots + (2n-1) = n^2$이 성립함을 증명하자.
  - $P(n) = 1 + 3 + 5 + \cdots + (2n-1) = n^2$이 성립하는가?
  - $n_0 = 1$

- $P(1) : 1 = 1^2$이므로 자명
- $P(k) : 1 + 3 + 5 + \cdots + (2k-1) = k^2$ 성립한다고 가정하자.
  - 양변에 $2k+1$을 더하면
  - $1 + 3 + 5 + \cdots + (2k-1) + (2k+1) = k^2 + 2k + 1 = (k+1)^2$
  - $1 + 3 + 5 + \cdots + (2k-1) + (2(k+1)-1) = (k+1)^2$
  - $1 + 3 + 5 + \cdots + (2(k+1)-1) = (k+1)^2 : P(k+1)$

강한 수학적 귀납법

- (base case) $P(n_0)$이 성립
- (inductive step) $n \le k$인 모든 n에 대해 $P(n)$이면 $P(k+1)$

- $n \ge n_0$인 모든 정수 n에 대해 $P(n)$이 성립
  - $P(n_0) \Rightarrow P(n_0+1)$
  - $P(n_0), P(n_0+1) \Rightarrow P(n_0+2)$
  - $P(n_0), P(n_0+1), P(n_0+2) \Rightarrow P(n_0+3)$

강한 수학적 귀납법

- 모든 트리에는 차수가 1 이하인 정점이 있음을 증명
  - P(n) := 정점이 n개인 트리에서 성립하는가?
  - n_0 = 1
- P(1) : 정점이 1개인 트리는 차수가 0인 정점이 있음
- 정점이 1..k인 트리에 모두 차수가 1 이하인 정점이 있다고 하자.
  - 정점이 k+1인 트리는 정점이 k개 이하인 1개 이상의 트리가
  - 새로 추가된 정점과 연결되어 있는 형태
  - 정점이 k개 이하인 트리에 차수가 1 이하인 정점이 있으므로
  - 정점이 k+1개인 트리에도 차수가 1 이하인 정점이 있음 : P(k+1)

질문?

분할 정복

분할 정복

- 분할 정복
  - 큰 문제를 여러 개의 작은 문제로 쪼갠 다음 (분할)
  - 작은 문제들의 답을 이용해 큰 문제의 답을 구함 (정복)

  - 동적 계획법?
    - 뒤에 있는 내용을 보면서 어떤 점이 다른 지 직접 확인해 보자.

분할 정복

- 분할 정복
  - 문제의 크기가 충분히 작은 경우 직접 해결
  - 문제의 크기가 큰 경우 K≥1개의 부분 문제로 쪼개서 각각 해결

```
void DivideAndConquer(InputType in, OutputType out){
    // 문제의 크기가 충분히 작은 경우 직접 해결
    if(in.size() <= Small){
        DirectSolve(in, out);
        return;
    }

    // 문제를 K개의 부분 문제로 분할함
    InputType in_small[K] = Divide(in, K);
    OutputType out_small[K];
    for(int i=0; i<K; i++){
        DivideAndConquer(in_small[i], out_small[i]);
    }
    out = Combine(out_small[0], out_small[1], ... , out_small[k-1]);
}
```

분할 정복

- 분할 정복의 시간 복잡도
  - $T(N) = D(N) + \text{sum } T(i) + C(N)$ if $N > \text{Small}$
$S(N)$ if $N \le \text{Small}$
    - N : 입력의 크기
    - T(N) : 입력의 크기가 N인 문제를 풀 때 걸리는 시간
    - D(N) : 크기가 N인 문제를 부분 문제로 분할할 때 걸리는 시간
    - C(N) : 크기가 N인 문제에서 부분 문제의 답을 합칠 때 걸리는 시간
    - S(N) : 크기가 N인 문제를 직접 해결할 때 걸리는 시간

  - 부분 문제들의 크기를 균등하게 분할하면
    - $T(N) = D(N) + aT(N/b) + C(N)$

질문?

분할 정복의 예시

분할 정복의 예시 – 1

- 정수 거듭제곱
  - 음이 아닌 정수 a, b, c에 대해 a^b(mod c)를 구하는 문제

  - b = 0이면 직접 해결 (a^b=1)
  - b ≥ 1이면 더 작은 문제로 나눠서 해결
    - b가 짝수면 $a^{b/2} * a^{b/2}$
    - b가 홀수면 $a^{(b-1)/2} * a^{(b-1)/2} * a$

  - 중복된 호출을 제외하면, 크기가 b인 문제를 크기가 b/2인 부분 문제 하나로 쪼갬
  - T(N) = O(1) + T(N/2) + O(1) = T(N/2) + O(1)
  - T(N) = O(log N)

분할 정복의 예시 – 2

- 합병 정렬
  - 배열의 구간 [l, r]을 정렬하는 알고리즘

  - l = r이면 직접 해결 (원소가 하나인 배열은 이미 정렬되어 있음)
  - l < r이면 더 작은 문제로 나눠서 해결
    - m = floor((l+r)/2)라고 할 때
    - [l, m]과 [m+1, r]을 각각 정렬한 다음 (분할)
    - 정렬된 두 배열을 합침 (정복)

  - 크기가 r – l인 문제를 크기가 약 (r-l)/2인 부분 문제 2개로 쪼갬
    - T(N) = O(1) + 2T(N/2) + O(N) = 2T(N/2) + O(N)
    - T(N) = O(N log N)

분할 정복의 예시 – 2

- 합병 정렬

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

int N, A[1010101];

void Merge(int s, int m, int e){
    static int tmp[1010101];

    int i = s, j = m + 1, idx = s;
    while(i <= m && j <= e){
        if(A[i] < A[j]) tmp[idx++] = A[i++];
        else tmp[idx++] = A[j++];
    }
    while(i <= m) tmp[idx++] = A[i++];
    while(j <= e) tmp[idx++] = A[j++];
    for(int k=s; k<=e; k++) A[k] = tmp[k];
}

void MergeSort(int s, int e){
    if(s == e) return;
    int m = (s + e) / 2;
    MergeSort(s, m);
    MergeSort(m+1, e);
    Merge(s, m, e);
}

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N;
    for(int i=1; i<=N; i++) cin >> A[i];
    MergeSort(1, N);
    for(int i=1; i<=N; i++) cout << A[i] << "\n";
}
```

질문?

분할 정복의 예시 – 3

- Inversion Counting
  - i < j & A[i] > A[j]를 만족하는 순서쌍 (i, j)의 개수를 세는 문제 (BOJ 1517 버블 소트)

  - 각각의 A[i]에 대해, 자신보다 뒤에 있으면서 자신보다 작은 A[j]의 개수를 세야 함
    - 단순하게 세면 $O(N^2)$
    - 원소들의 실제 위치가 아닌 전후 관계만 중요하다는 것에 주목하자.

분할 정복의 예시 – 3

- Inversion Counting
  - 배열을 반으로 잘라보자.
    - [l, m]과 [m+1, r] 안에서 발생하는 inversion은 재귀적으로 계산하고 (분할)
    - [l, m]과 [m+1, r] 간의 inversion의 개수만 세면 된다. (정복)
    - [l, m]의 원소 x보다 작은 [m+1, r]의 원소 y의 개수를 세면 된다.
      - 합병 정렬과 똑같은 방법으로 할 수 있음

- $T(N) = O(1) + 2T(N/2) + O(N) = 2T(N/2) + O(N)$
- $T(N) = O(N \log N)$

```
long long R = 0;

void Merge(int s, int m, int e){
    static int tmp[505050];

    int i = s, j = m + 1, idx = s, cnt = 0;
    while(i <= m && j <= e){
        if(A[i] <= A[j]) tmp[idx++] = A[i++], R += cnt;
        else tmp[idx++] = A[j++], cnt += 1;
    }
    while(i <= m) tmp[idx++] = A[i++], R += cnt;
    while(j <= e) tmp[idx++] = A[j++];
    for(int k=s; k<=e; k++) A[k] = tmp[k];
}
```

질문?

분할 정복의 예시 – 4

- BOJ 1725 히스토그램
  - 히스토그램에서 가장 큰 직사각형을 구하는 문제
    - 배열에서 (j-i+1) * min(A[i], A[i+1], … , A[j])의 최댓값을 구하는 문제

  - 단순하게 계산하면 $O(N^3)$
  - 조금 더 똑똑하게 계산하면 $O(N^2)$

  - 배열을 단순히 반으로 나누는 것은 조금 애매함…

분할 정복의 예시 - 4

- BOJ 1725 히스토그램
  - 배열을 반으로 잘라보자.
    - A[m]을 포함하는 모든 구간을 처리하면
    - 고려하지 않은 구간은 [l, m-1] 또는 [m+1, r]에 완전히 포함됨

    - [l, m-1]과 [m+1, r]에 포함된 구간은 재귀적으로 잘 처리하고
    - [l, r]에서 A[m]을 포함하는 모든 구간을 확인하는 방법만 찾으면 됨

분할 정복의 예시 – 4

- BOJ 1725 히스토그램
  - A[m]을 포함하는 모든 구간을 확인
    - 단순하게 확인하면 $O(N^2)$ / 더 빠르게 해야 함

    - [m, m]부터 시작해서 한 칸 씩 확장하는 방식으로 진행
    - 왼쪽으로 확장해야 할까? 오른쪽으로 확장해야 할까?

    - 확장 방향에 관계없이 (j-i+1)의 값은 동일하게 1 증가함
    - min(A[i], A[i+1], … , A[j])가 덜 작아지는 방향으로 확장해야 함
    - 확장은 r-l번 하므로 선형 시간에 확인할 수 있음
  - $T(N) = 2T(N/2) + O(N)$
  - $T(N) = O(N \log N)$

```cpp
using ll = long long;

ll N, A[1010101];

ll Solve(int s, int e){
    if(s > e) return 0;
    if(s == e) return A[s];
    int m = (s + e) / 2;

    ll res = A[m], cnt = 1, mn = A[m];
    int i = m - 1, j = m + 1;

    while(s <= i && j <= e){
        if(A[i] > A[j]) cnt++, mn = min(mn, A[i--]);
        else cnt++, mn = min(mn, A[j++]);
        res = max(res, cnt * mn);
    }
    while(s <= i){
        cnt++; mn = min(mn, A[i--]);
        res = max(res, cnt * mn);
    }
    while(j <= e){
        cnt++; mn = min(mn, A[j++]);
        res = max(res, cnt * mn);
    }
    return max({ res, Solve(s, m-1), Solve(m+1, e) });
}
```

질문?

과제

- 필수
  - 1629 곱셈
  - 2751 수 정렬하기 2 (합병 정렬 직접 구현)
  - 2630 색종이 만들기
  - 1517 버블 소트
  - 1725 히스토그램
- 심화
  - 21870 시철이가 사랑한 GCD
  - 2261 가장 가까운 두 점