CSE3081-1: Design and Analysis of Algorithms (Fall 2018)

Machine Problem 1: Maximum Subsequence Sum

Report

전공 : 컴퓨터공학과

학번 : 20171667

이름 : 이용욱

보고서 목차

1. 과제 개요
2. 작성한 코드에 대한 설명
3. 알고리즘의 종류와 입력의 크기에 따른 소요시간
4. 프로젝트 개요

정수로 이루어진 수열이 주어졌다. 이 수열의 부분수열의 원소의 합 중 가장 큰 것을 Maximum Subsequence Sum (최대 부분수열 합)이라고 한다. 우리 과제에서는 최대 부분수열 합을 구하는 세가지 알고리즘을 C/C++ 로 구현해보고, 알고리즘의 종류와 입력의 크기에 따른 소요시간을 비교해보았다.

1. 작성한 코드

C++로 작성하였으며 Visual Studio 2017에서 개발하였다.

G++ 컴파일러에서도 정상적으로 작동함을 확인하였다.

* 1. main (int argc, char \*argv[]) 함수

매개변수로 입력파일의 파일명과 알고리즘의 인덱스를 받는다.

입력/출력 파일을 열고 인덱스에 해당하는 알고리즘을 구현하는 함수에 이를 넘겨준다

* 1. mss\_bf (FILE\* in, FILE\* out) 함수

2중 for문으로 i항부터 j항까지의 누적 합을 계산한다.

이 때 현제의 mss보다 큰 누적 합을 찾으면, 현제의 누적합이 mss이고, I, j가 mss의 양 끝이 된다.

이때 < 비교연산자로 비교를 해 주면 mss 앞의 0은 포함되고 뒤의 0은 제외된다.

걸린 시간은 <ctime>의 clock()을 활용하였다. 시작과 끝의 클럭을 빼고 초당 클록(CLOCKS\_PER\_SEC)으로 나눠주면 걸린 시간을 구할 수 있다. 1000을 곱하면 단위를 ms으로 할 수 있다.

시간복잡도는 O(N^2) 이다.

* 1. mss\_dc (FILE\* in, FILE\* out) 함수

mss를 divide and conquer 알고리즘으로 구한다. 제귀함수를 divide\_conquer()라는 함수로 따로 구현했으므로 divide\_conquer(0, n-1, mss, sub\_start, sub\_end); 를 호출한다.

시간복잡도는 F\_n = F\_n/2 + O(N)을 계산하면, O(NlogN) 이다.

* 1. divide\_conquer (int s, int e, int& mss, int& sub\_start, int &sub\_end) 함수

[s, e]를 [s, mid], [mid+1, e]로 나누어 정복한다.

그러기 위해 다음 두 함수를 호출한다.

divide\_conquer(s, mid, mss\_left, start\_left, end\_left);

divide\_conquer(mid+1, e, mss\_right, start\_right, end\_right);

BaseCase는 e==s일 때이다. e와 s가 같은경우 배열의 한 원소만을 보는 것 이므로 mss = arr[s]이다.

두 영역을 모두 포함하는 부분배열이 더 큰 mss를 가질 수 있다. 따라서 mid, mid+1을 포함하는 mss를 구한다. 우선 mid+1에서 시작하여, 오른쪽으로 배열를 누적해서 더하며 mss를 갱신한다. 그 이후 mid에서 시작하여, 왼쪽으로 배열를 누적해서 더하며 mss를 갱신한다.

이후 가장 mss가 큰 것을 int& mss, int& sub\_start, int &sub\_end 에 저장하고 함수를 종료한다.

1. 알고리즘의 종류와 입력의 크기에 따른 소요시간
   1. 실험환경

구동환경

OS : Windows 10 Education 64bit

컴파일 및 실행 : MicroSoft 사의 Visual Studio 2017

구동 PC

CPU : Intel i5 – 7600 (3.5GHz)

RAM : 8.00GB

* 1. 실험 설정

위에서 구현한 세 알고리즘이 실제로 다양한 크기의 입력을 처리하는 시간을 비교하였다.

입력은 수열의 길이를 10부터 10의 배수로 늘려가며 10^6 까지 입력하였다. 입력파일은 rand()함수를 이용하여 절댓값이 1000이하인 임의의 정수배열을 생성하였다.

지나치게 특수한 입력이 존재할 수도 있으므로 입력은 한 크기 당 2개의 입력 파일을 만들었고, 평균값을 결과로 채택하였다.

* 1. 실험 결과

그 결과는 다음과 같았다

입력이 10^6 일 때 O(n^2) algorithm은 실험할 수 없었다.

실험 PC는 약 1초에 10^9번 연산하는데 n=10^6 일때 n^2 = 10^12 이어서 실행하는 데에 수천 초 (수 시간) 이 필요하기 때문이다.

|  |  |  |  |
| --- | --- | --- | --- |
|  | O(n) | O(nlogn) | O(n^2) |
| 10^1 | 0.002000 | 0.004000 | 0.003000 |
| 10^2 | 0.003000 | 0.018000 | 0.037000 |
| 10^3 | 0.010000 | 0.174000 | 3.194000 |
| 10^4 | 0.079000 | 1.800000 | 214.483000 |
| 10^5 | 0.652000 | 17.373000 | 13415.527000 |
| 10^6 | 3.656000 | 119.121000 | - |

그레프로 나타내면 다음과 같았다.

* 1. 실험결론

이 결과가 실제로 시간복잡도와 유사한 지 확인하기 위해 시간복잡도로 나누어 보았다



같은 알고리즘끼리는 입력이 10배가 되어도 실행시간을 시간복잡도로 나눈 값 크게 차이 나지 않았다. 다만 이 값은 입력이 커질수록 감소하였는데, 이는 변수할당, 함수호출 등의 작업은 모든 크기의 입력에 대해 똑같이 이루어짐을 고려하면 납득할 수 있었다.

결론적으로, 우리의 알고리즘의 실행시간은 해당 알고리즘의 시간복잡도를 따랐다고 할 수 있다.

1. 배운 점

우선 MSS문제를 해결하는 세 알고리즘과 그 구현을 배웠다.

이 과정에서 string 객체나, time모듈에 관해서도 알게 되었다.

문제를 해결할 때, 더 효율적으로 해결하기 위해 생각하는 방법도 배웠다.