CSE3081 (1반): 알고리즘 설계와 분석
[숙제 1] Detection of Brightest Area in 2D Image

담당 교수: 임 인 성

2024년 9월 12일

마감: 9월 29일 일요일 오후 8시 정각
제출물, 제출 방법, LATE 처리 방법 등: 조교가 과목 게시판에 공지함.

목표: 이번 숙제는 주어진 문제에 대하여 서로 다른 시간 복잡도를 가지는 알고리즘을 구현한 후, 다양한 크기의 입력 데이터에 대한 수행 시간을 측정하여, 이론적인 시간 복잡도와 실제 수행 시간 간의 연관 관계를 분석해봄을 목표로 한다.

문제

1. 다음과 같은 Maximum Sum Subarray Problem (1D)을 고려하자.

   Given a 1D array of size n, find the maximum-sum subarray **with at least one element**.

   이 문제를 해결해주는 다음과 같은 시간 복잡도를 가지는 두 가지 알고리즘을 구현하라. 각 방법은 최대 합뿐만 아니라 그에 해당하는 subarray의 처음과 마지막 원소의 인덱스를 찾아주어야 한다.

   - **Algorithm 1**: 시간 복잡도 $O(n \log n)$ ⟵ Divide-and-conquer 기법을 적용한 방법
   - **Algorithm 2**: 시간 복잡도 $O(n)$ ⟵ Dynamic programming 기법을 적용한 방법 (Kadane's algorithm)

2. 다음과 같은 Maximum Sum Subrectangle Problem (2D)을 고려하자.

   Given a 2D integer array of size $n \times n$, find the maximum-sum subrectangle **with at least one element**.

   이 문제를 해결해주는 다음과 같은 시간 복잡도를 가지는 세 가지 알고리즘을 구현하라. 각 방법은 최대 합뿐만 아니라 그에 해당하는 subrectangle의 위-왼쪽 모서리와 아래-오른쪽 모서리 원소들의 인덱스를 찾아주어야 한다.

   - **Algorithm 3**: 시간 복잡도 $O(n^4)$ ⟵ Summed Area-Table 기법 적용 방법 (강의 설명)
   - **Algorithm 4**: 시간 복잡도 $O(n^3 \log n)$ ⟵ 1D 문제를 풀기 위하여 Algorithm 1을 적용한 방법
   - **Algorithm 5**: 시간 복잡도 $O(n^3)$ ⟵ 1D 문제를 풀기 위하여 Algorithm 2를 적용한 방법 (강의 설명)

3. 이번 숙제의 목적은

   (a) 2D 문제를 서로 다른 시간 복잡도를 가지는 Algorithm 3, Algorithm 4, 그리고 Algorithm 5를 구현하여,
   (b) 서로 다른 크기 n을 가지는 입력 영상에 대하여 수행 시간을 측정한 후,
   (c) 과연 그러한 수행시간이 이론적인 시간 복잡도와 일치하는지를 확인하는 것이다.

입출력 방식

1. 이번 숙제에서 구현하는 프로그램은 Visual Studio 솔루션의 프로젝트 디렉토리의 `Data` 디렉토리에 존재하는 **이름이** `HW1_config.txt`**인 텍스트 파일**에서 먼저 테스트 케이스의 개수를 읽어들인 후, 한번에 한 줄씩 (i) 사용할 알고리즘 번호, (ii) 입력 영상 파일 이름, (iii) 입력 영상의 평균값 파일 이름, 그리고 (iv) 그에 대한 결과 값 출력 파일 이름을 순서대로 읽어 들여, 해당하는 알고리즘을 사용하여 그에 대한 계산을 수행한 후, 출력 파일에 계산 결과를 출력해주어야 한다. 이때 입출력 파일은 모두 위의 `Data` 디렉토리에 존재하여야 하며, 다음은 `HW1_config.txt` 파일의 예이다.

```
9
3 moon_64.pgm moon_64_average.txt moon_64_brightest_3.txt
4 moon_64.pgm moon_64_average.txt moon_64_brightest_4.txt
5 moon_64.pgm moon_64_average.txt moon_64_brightest_5.txt
3 moon_128.pgm moon_128_average.txt moon_128_brightest_3.txt
4 moon_128.pgm moon_128_average.txt moon_128_brightest_4.txt
5 moon_128.pgm moon_128_average.txt moon_128_brightest_5.txt
3 moon_256.pgm moon_256_average.txt moon_256_brightest_3.txt
4 moon_256.pgm moon_256_average.txt moon_256_brightest_4.txt
5 moon_256.pgm moon_256_average.txt moon_256_brightest_5.txt
```

2. 이번 숙제에서 해결하려는 2D 문제의 첫 번째 파일인 입력 영상 파일 (위의 예에서 `***_n.pgm`)은 pgm 파일 포맷으로 저장되어 있으며 (여기서 `n`은 영상 데이터의 해상도 `n` $\times$ `n`을 의미함), 두 번째 파일인 평균값 파일 (위의 예에서 `***_n_average.txt`)에는 입력 영상의 모든 픽셀의 광도의 평균이 정수 값으로 기록되어 있다. **먼저 pgm 형식의 영상을 2D 배열로 읽어들인 후 각 픽셀의 광도값에서 평균값을 뺀 2D 영상을 자신이 구현한 알고리즘의 입력 영상으로 사용하라**.

3. 한편, 세 번째 파일인 출력 파일 (위의 예에서 `***_n_brightes_m.txt`)에는 계산 수행 결과를 기술하는 다섯 개의 정수 값들이 ASCII 형태로 저장되어야 한다. (여기서 `m`은 사용한 알고리즘 번호임.)

   - $s\ k\ i\ l\ j$

   여기서 s는 여러분이 찾 subrectangle의 sum이고, k와 i는 그 subrectangle의 위-왼쪽 모서리의 인덱스, 그리고 l와 j은 아래-오른쪽 모서리의 인덱스를 나타낸다 (강의자료 1의 61쪽 그림 참조). 입력 데이터의 인덱스는 각 방향 0부터 시작하며, 만약 동일한 최대 합을 가지는 subrectangle이 여러 개가 있을 경우, 어떤 것에 대한 정보를 출력해도 무방하나, 가급적 $i \rightarrow j \rightarrow k \rightarrow l$ 순서대로 인덱스가 가장 작은 subrectangle에 대한 인덱스를 출력하도록 하라.

채점 내용

1. 보고서의 가장 첫 부분에 세 개 알고리즘 (Algorithm 3, Algorithm 4, 그리고 Algorithm 5) 각각에 대하여 자신이 사용한 방법을 몇 줄 이내의 문장으로 간략히 요약하라. 만약 세 개 모두 구현을 하지 못했다면, 어떤 알고리즘을 구현하였는지를 **정확히** 밝힐 것.

2. 여러분의 프로그램은 원하는 maximum sum subrectangle을 정확히 찾아주어야 한다. 채점은 제공한 두 부류의 데이터 외에 자체적으로 생성한 입력 데이터를 사용하여 여러분의 프로그램이 정확한 결과를 산출하는지 **기계적으로 확인**할 예정임. **따라서 입출력 파일의 형식에 문제가 있을 경우 0점 처리가 됨**.

3. 다음과 같이 실제 수행 시간에 대한 실험을 진행한 후, 그 결과를 보고서에 기술하라.

   (a) 수행 시간 및 시간 복잡도의 관계를 충실히 분석할 수 있도록, 주어진 입력 크기 n에 대해 위 두 문제에 대한 실험을 진행하라. 시간 측정은 **x64 플랫폼**에서 **Release 모드**로 컴파일하여 ms 단위로 측정하라. 특히 실험 결과의 신뢰도를 높이기 위하여 가급적 동일한 입력 영상에 대하여 여러 번 수행한 후 평균 시간을 취할 것.

(b) 상기 실험을 통하여 산출한 실험 결과로부터 세 알고리즘(Algorithm 3, Algorithm 4, 그리고 Algorithm 5)의 이론적인 시간 복잡도와 수행 시간 간의 관계를 분석하고, 그로부터 자신이 발견한 사실과 그에 대한 의견을 보고서 파일에 **명확히** 기술하라.

- 본 숙제의 중요한 목적 중의 하나는 이론적인 시간 복잡도와 실제 프로그램의 수행 시간과의 연관성을 찾아내는 것인데, 각 알고리즘에 대한 실험 결과를 **적절한 형식의 테이블로 요약**하라. 실제 측정한 시간 값과 이론적으로 구한 시간 복잡도의 연관성을 증명하기 위하여 그래프를 그린다던가 또는 수식으로 함수 관계를 보인다던가 하는 등의 공학적인 방법을 사용하여 자신의 주장을 논리적으로 기술할 것: 충분히 큰 n에 대하여 $O(n^4)$ 방법과 $O(n^3 \log n)$ 방법의 차이를 어떻게 확인할 수 있을까?
- 보고서에 실험 결과를 기술하기 전에 본인이 사용한 컴퓨터의 실험 환경을 정확히 기술할 것 (아래의 예를 참조할 것).

```
OS: Windows 11 Education
CPU: Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz
RAM: 16.00GB
Compiler: Visual Studio 22 Release Mode/x64 Platform
```

주의 사항

1. 숙제 제출 방법에 대하여 조교가 사이버 캠퍼스에 공지하는 내용을 숙지할 것.
2. 채점 시 형식 상의 문제(예를 들어, configuration 파일의 내용을 정확히 읽어들이지 못할 경우) 테스트 데이터에 대하여 답을 찾지 못한 것으로 간주할 예정임: 다수의 학생에 대하여 기계적으로 채점을 해야하니 양해 바람.
3. 시간 측정은 본 수업에서 배포하는 예제 프로그램에서 사용한 시간 측정 방법을 이해하여 사용할 것.
4. 제출한 원시 코드에서 대해서는 copy-check를 수행할 예정이며, 다른 사람의 코드 또는 보고서를 복사할 경우 **관련된 사람 모두에 대하여 (즉 복사한 사람과 복사 당한 사람 모두)** 과목 최종 성적의 50%를 감점함.