알고리즘 H/W #2

Due: 10/21(수) 3:00pm
제출: 302-313-2(최적화연구실)
조교: 이상엽(880-1851)

● 수행 시간에 대한 요구 사항이 명시되지 않은 문제는 최대한 효율적인 알고리즘을 찾아야 한다. 효율성이 떨어지는 정도에 비례해서 점수를 적게 받게 됨. 모든 문제 공히 여러분의 알고리즘에 대해 수행 시간을 밝히고 그렇게 되는 이유를 설명해야 한다. 여러분이 제대로 알고 말하고 있다는 것을 채점자가 알 수 있는 수준이면 됨. 점수는 알고리즘 75%, 수행 시간 설명 25%. 구체적인 알고리즘을 쓰지 않고 Optimal substructure만 제대로 밝혔다면 75% 중 60%p 부여. 즉, Optimal substructure를 밝히고 수행 시간 설명을 제대로 했다면 85%의 점수를 부여함.

1. (20점) 앞에서부터 읽거나 뒤에서부터 읽거나 똑같은 string을 palindrome이라고 한다. {A, B, C, D}로 이루어진 임의의 string의 모든 substring 중 가장 긴 palindrome의 길이를 알아내는 알고리즘을 dynamic programming으로 작성하라. 예를 들어, string ABBCDAA에는 AA, ABBA, ABA, ACA, ADA, A, B, C, D 등의 substring이 palindrome을 이룬다. 가장 긴 palindrome의 길이는 4이다.

2. (20점) 길이가 n이고 3 줄짜리 놀이판이 주어지고 놀이판의 각 칸에는 양의 정수가 쓰여져 있다. 이 놀이판의 모든 칸을 ○, △, ×를 써서 표시하는데 매 세로열마다 세 가지 표시를 다 사용해야 하고 (즉, ○, △, ×가 각각 한 번씩 나타나야 하고), 가로로 이웃하는 두 칸에는 같은 표시가 나타나면 안된다. 이렇게 하면 세 가지 표시가 각각 n 번씩 나타나게 된다.

   모든 칸에 대해 표시를 마치면 이에 대해 점수를 매기는데, ○ 표시된 칸의 숫자는 더해주고, × 표시된 숫자는 빼주고, △ 표시된 숫자는 무시한다 (0으로 처리). 아래 그림은 한 예를 보인다. 첫 번째는 주어진 놀이판, 두 번째는 이 놀이판에서 한 가지 표시 방법, 세 번째는 위의 두 결합(놀이판과 한 표시 방법)에 의한 각 칸의 계산 수치를 나타낸다. 즉, 이 경우의 총 점수는 2 - 10 + 13 - 6 + 11 - 9 - 15 + 16 ... 와 같이 된다.

2	8	11	15	9	28	19	16	41	34	28	9
10	13	9	16	20	18	32	26	15	37	24	3
5	6	7	16	25	31	21	29	39	29	19	10

○	△	○	×	○	×	△	○	△	○	×	○
×	○	×	△	×	△	○	×	○	×	△	×
△	×	△	○	△	○	×	△	×	△	○	△

⬇

+2		+11	-15	+9	-28		+16		+34	-28	+9
-10	+13	-9		-20		+32	-26	+15	-37		-3
	-6		+16		+31	-21		-39		+19	

길이 n인 임의의 놀이판이 주어졌을 때 받을 수 있는 최고 점수를 찾아내는 complexity가 $O(n)$인 알고리즘을 dynamic programming에 의해 제시하라.

3. (20점) 주어진 n 개의 서로 다른 실수의 배열로부터 longest monotonic increasing subsequence를 찾아내는 $O(n^2)$ 알고리즘을 dynamic programming으로 작성하여라. 예를 들어, 배열 11, 17, 5, 8, 6, 4, 7, 12, 3이 주어질 때 longest monotonic increasing subsequence는 5, 6, 7, 12이다.

4. (20점) 영문 전용 워드 프로세서에서 한 문단을 보기 좋게 만드는 방법에 대해 생각해 보자. 한 문단은 n 개의 단어로 이루어지고 여기서 한 단어란 붙여쓰는 글자 묶음을 말한다. 예를 들어 "We are attacking an interesting problem."이라는 문장은 6 개의 단어로 이루어져 있다고 말한다. 문제를 간명하게 하기 위해 다음과 같이 몇 가지 가정을 하자.

   a. 한 character가 차지하는 넓이는 모두 똑같다 (편의상 마침표도 똑같은 넓이를 차지한다고 가정해도 좋음).
   b. 단어와 단어 사이에는 한 칸 (한 character 넓이만큼) 띄운다. 단, 줄이 바뀔 때는 띄울 필요 없다.
   c. 한 단어는 잘라져 두 줄에 걸칠 수 없다.

   한 줄은 character 80 개가 들어갈 수 있는 넓이이다. 임의의 문단에 들어가는 n 개의 단어들의 길이(character수)를 각각 $w_1, w_2, \ldots, w_n$이라 할 때, i번째 단어부터 j번째 단어까지가 한 줄에 들어간다고 하면, 이를 위해 필요한 최소한의 공간은 $\sum_{k=i}^{j} w_k + j - i$ 가 되고, $80 - j + i - \sum_{k=i}^{j} w_k$ 만큼의 공간은 여백으로 남는다 (문제의 간명성을 위해 오른쪽을 여백으로 남기는 것으로 하자).

   한 문단의 단어들을 각 줄에 할당하는 방법은 수없이 많은데 위에서 언급한 추가적인 여백의 양을 최소화하는 할당 방법을 찾고자 하는 것이 본 문제의 목적이다 (단, 마지막 줄의 여백은 미관을 해치지 않으므로 고려하지 않는다). n 개의 단어로 된 문단이 주어질 때 여백의 "제곱합"의 최소값을 찾아내는 linear-time dynamic programming 알고리즘을 만들어 보라. Running-time이 linear time임도 간단히 설명할 것.

5. (20점) A[1][1] ~ A[N][N] 행렬에 -100부터 100 사이의 임의의 수들이 채워져 있다. 이 행렬의 A[1][1]부터 시작하여 인접한 원소들을 방문해가면서 A[N][N]에 이르는 경로를 하나 선택하려 한다. A[1][1]은 가장 위쪽열의 가장 왼쪽 원소를 가리킨다. 단, N은 500을 넘지 않는다. 방문 패턴에는 아래와 같은 제약이 있다.

   1. 위쪽으로는 이동할 수 없다.
   2. 한 번 방문한 원소는 다시 방문할 수 없다.

   이러한 제약을 만족하면서 임의의 경로를 따라 A[N][N]에 이르면 이 경로상에 방문되었던 원소들의 값의 총합이 그 경로의 점수가 된다. 임의의 N*N 행렬이 주어질 때, A[1][1]에서 A[N][N]에 이르는 경로 중 가장 점수가 높은 경로의 점수를 찾는 $O(N^2)$ 알고리즘을 작성하라.

아래에 5*5 행렬을 하나 예로 들어 설명한다. 그림 1은 가능한 방문 패턴의 예를 두 개 보인다.

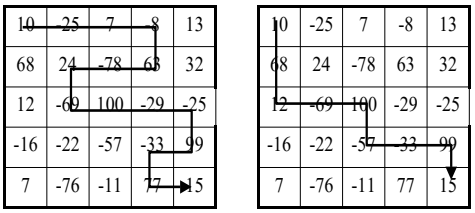

그림 1: 적합한 경로들의 예

그림 2는 가능하지 않은 방문 패턴의 예를 두 개 보인다. 그림2의 첫 번째 예는 위쪽으로 이동함으로서 규칙을 어긴 것이고, 두 번째 예는 한 번 방문한 원소를 다시 방문함으로써 규칙을 어긴 것이다.

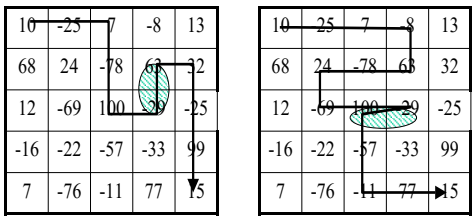

그림 2: 부적합한 경로들의 예