**알고리즘 설계 및 분석 HW3**

컴퓨터공학과 20171666 이예은

1. Gapped-alignment problem
2. 구현 시 사용한 변수

|  |  |
| --- | --- |
| x | 첫번째 문자열 |
| Y | 두번째 문자열 |
| Xsize | X 문자열의 길이 |
| Ysize | Y 문자열의 길이 |
| Gap\_x | Gap을 포함한 x 문자열 |
| Gap\_y | Gap을 포함한 y 문자열 |
| Gap | Gap을 계산하기 위해 만든 배열로 이전에 최대값을 어디서 가져왔는지를 저장해준다. 1인 경우 대각선으로 최대값을 받은 것이고, 2인 경우 i-1,j에서, 3인경우 I,j-1에서 최대값을 가져온 것이다. |
| Gap\_xsize | X 문자열의 gap의 개수 |
| Gap\_ysize | Y 문자열의 gap의 개수 |
| Dp\_table | Dynamic programming을 하는데 필요한 x\*y 크기만큼의 배열 |
| X\_point | Gap을 포함한 문자열을 만드는데 사용한 변수로 x 문자열의 적절한 위치의 인덱스 |
| Y\_point | Gap을 포함한 문자열을 만드는데 사용한 변수로 y 문자열의 적절한 위치의 인덱스 |
| Newx\_point | Gap을 포함한 문자열을 만드는데 사용한 변수로 gap\_x가 어디까지 채워져 있는지를 나타내는 인덱스 |
| Newy\_point | Gap을 포함한 문자열을 만드는데 사용한 변수로 gap\_y가 어디까지 채워져 있는지를 나타내는 인덱스 |
| Match | Dp 함수에서 대각선 방향에서 올 때의 점수. Dp\_table[i-1][j-1]+is\_match()이다. |
| Ingap\_x | Dp 함수에서 위쪽 방향에서 올 때의 점수. Dp\_table[i-1][j]-p이다 |
| Ingap\_y | Dp 함수에서 왼쪽 방향에서 올 때의 점수. Dp\_table[i][j-1]-p이다 |
| maxitem | Match, ingap\_x, ingap\_y중 가장 큰 값 |

1. 구현 방법
2. Main() : 메인 함수에서는 우선 파일들을 열고 안에 있는 정보를 적절히 알맞은 배열에 입력받는다. 이 후 dp() 함수를 호출하여 dynamic programming 방식으로 문제를 해결한다. Dp 가 끝났다면 main에는 dp\_table과 gap 배열이 제대로 차있을 것이며 이를 이용하여 gap의 위치와 크기를 구한다. 구하는 방식은 다음과 같다. 우선 배열의 가장 마지막, gap[xsize][ysize] 부터 검사하기 시작한다. Gap 값이 1이라면 대각선으로 왔고, gap이 삽입되지 않은 것이므로 gap\_x와 gap\_y에 각각 원래 본인 문자를 넣는다. 그 후 gap[i-1][j-1]로 이동하여 다시 검사한다. Gap 값이 2라면 위에서 온 것이고 y에 gap이 삽입된 것이므로 gap\_x에는 원래 본인 문자를, gap\_y에는 ‘\_’를 넣는다. 그 후 gap[i-1][j]로 이동한다. Gap이 3이라면 왼쪽에서 왔고, x에 gap이 삽입된 것으로 gap\_x에 ‘\_’를, gap\_y에 원래 문자를 넣는다. 그 후 gap[i][j-1]로 이동한다. 이 과정을 gap[0][0]에 도달할 때까지 반복하면 gap\_x와 gap\_y에 gap이 포함된 각각의 문자열이 거꾸로 삽입되게 된다. 이후에는 이 두 문자열을 각각 돌면서 gap이 어느 인덱스에 있는지 확인할 수 있으므로 ‘\_’가 나올 때마다 위치를 출력해준다. 그리고 프로그램이 종료된다.
3. Dp() : dp 함수에서는 우선 dp\_table의 첫번째 행과 열을 초기화해준다. 초기화는 앞자리에 전부 gap을 삽입한다 생각하는 것이기 때문에 -2i를 넣어준다. 또한 dp\_table[i][0]의 경우에는 전부 위쪽에서 오는 것이므로 gap[i][0] 배열에 2를 넣어주고, dp\_table[0][i]의 경우에는 전부 왼쪽에서 오는 것이므로 gap[0][i]에 3을 넣어준다. 이후 이중 for loop를 돌려 dp\_table을 전부 돌면서 최대 유사성 점수를 계산하기 시작한다. 최대 유사성 점수는 경우가 3가지로 나뉘는데, 한가지는 x나 y에 gap을 삽입하지 않고 두 문자를 비교해 match나 mismatch 점수를 주는 경우, x에 gap을 삽입하여 gap panalty를 받는 경우, y에 gap을 삽입하여 gap panalty를 받는 경우이다. 각각의 경우 중 가장 최대 점수를 dp\_table[i][j]에 삽입해야 한다. 첫번째 경우는 x,y가 모두 새로 추가되므로 대각선 이전의 위치에서의 dp\_table값에 match인지 mismatch인지의 점수를 알려주는 is\_match() 함수의 리턴값을 더한다. gap에는 1을 넣어준다. 두번째 경우는 y는 하나 늘고 x는 그대로인데 gap을 넣는 것이므로 위쪽의 dp\_table 값에 gap penalty 값을 더해준다. gap에는 2를 넣어준다. 세번째 경우는 x는 하나 늘고 y는 gap을 넣는 것이므로 왼쪽의 dp\_table 값에 gap panalty를 더해준다. gap에는 3을 넣어준다. 이 세가지 값 중 max를 찾아 dp\_table[i][j]에 넣어주고 다음으로 넘어간다. 이 반복문이 모두 끝나면 가장 마지막 자리, dp\_table[xsize][ysize] 자리에는 전체 최대값이 저장되있을 것이므로 이를 main으로 리턴해준다.
4. Is\_match() : x와 y 배열 및 검사하려는 위치를 받아 둘이 같으면 match 점수를, 다르면 mismatch 점수를 리턴한다.
5. Card problem
6. 구현 시 사용한 변수

|  |  |
| --- | --- |
| Right | 오른손에 쥔 카드의 배열 |
| Left | 왼손에 쥔 카드의 배열 |
| Left\_size | Left 배열 크기 |
| Right\_size | Right 배열 크기 |
| Card\_one | 검사하려는 첫번째 카드배열 |
| Card\_two | 검사하려는 두번째 카드배열 |
| Rpoint | Right 배열에서 검사하고 있는 숫자의 인덱스 |
| Lpoint | Left 배열에서 검사하고 있는 숫자의 인덱스 |
| R | Right와 검사했을 때의 점수 |
| L | Left와 검사했을 때의 점수 |
| Max | R와 l 중 최대값 |
| Dp\_table | Dynamic programming을 적용하기 위한 rsize+lsize+1 크기의 배열 |

1. 구현 방법
2. Main() : 메인 함수에서는 파일을 열고 데이터를 알맞은 배열에 저장한다. 이 후 dp() 함수를 불러 섞어서 만들 수 있는지 아닌지에 대한 결과를 얻고, 이를 출력해준다.
3. Dp() : dynamic programming을 하기 위한 함수이다. Dp 방법은 다음과 같다. 경우를 3가지로 나눌 수 있는데, 검사하려는 card 묶음의 해당 숫자가 right와 같은 경우, left와 같은 경우, 둘다 아닌 경우이다. Right와 같은 경우와 left와 같은 경우는 둘다 1점씩, 둘다 아닌 경우는 0점을 주도록 구현한다. Dp\_table의 이전 배열 값은 카드 묶음의 이전 숫자들이 잘 배열 되었는지에 대한 점수가 저장되있으므로 이 값에 경우에 따라 점수를 추가해주는 형식이다. 이를 코드로 구현해 보면, 우선 dp\_table의 가장 처음 자리 dp\_table[0]을 0으로 초기화해준다. 이는 맨 처음에는 아무런 점수를 주지 않기 위해서이다. 이후 for loop를 돌면서 배열을 채워 나간다. 채울 때에는 먼저 이전 dp\_table[i-1]의 결과에서 right과 match하면 1점, 아니면 0점을 더해주기 위해 match(right[rpoint],pack[i])의 반환값을 더해준 r을 구하고, 이전 dp\_table[i-1]의 결과에서 left와 match하면 1점, 아니면 0점을 더해주기 위해 match(left[lpoint],pack[i])의 반환값을 더해준 l을 구한다. 이 후 이 중 max 값을 구해 dp\_table[i]에 넣어준다. 이렇게 하면 right와 일치할 경우 right의 1점이 더해져 r이 최대값으로 선택되고, left와 일치할 경우 left의 1점이 더해서 l이 최대값으로 선택된다. 그러므로 dp\_table[i]에 이전결과 + 1 한 결과가 저장된다. 모두 아닌 경우라면 둘다 이전 결과와 같으므로 이전 결과를 현재 dp\_table[i]에 그냥 저장하게 된다. 또한 right와 일치한 경우 right의 다음 숫자를 검사하도록 rpoint를 ++ 해주고, left와 일치한 경우 left의 다음 숫자를 검사하도록 lpoint를 ++ 해준다. 모든 loop가 끝났으면 dp\_table의 가장 마지막 위치 dp\_table[rsize+lsize]를 검사한다. 만약 이 값이 rsize+lsize와 다르다면 중간에 한번이라도 둘다 아닌 경우가 생겨 0이 더해졌다는 것으로 결과값을 0을 리턴한다. 둘이 같다면 카드섞기가 가능한 묶음이므로 1을 리턴한다.
4. Match() : 두 개의 값을 받아서 만약 두개가 같을 경우 1을 리턴하고, 다를 경우 0을 리턴해주는 함수이다.