문제 01

동적 계획법에서 상태를 정의한다는 뜻으로 가장 적절한 것은?

1. 입력의 정렬 순서만 정한다
2. 각 반복에서 당장 가장 좋은 선택만 정한다
3. 배열이나 함수의 각 값이 어떤 부분 문제의 답인지 정한다
4. 배열 크기를 정하면 각 값의 의미는 필요 없다
5. 최종 답을 모든 상태의 초기값으로 복사한다

문제 01 정답과 해설

동적 계획법에서 상태를 정의한다는 뜻으로 가장 적절한 것은?

3번 배열이나 함수의 각 값이 어떤 부분 문제의 답인지 정한다

해설

상태의 의미가 정해져야 점화식, 초기값, 계산 순서를 일관되게 설계할 수 있다.

문제 02

동적 계획법이 중복 계산을 줄이는 방식은?

1 같은 인수라도 재귀 경로마다 새로 계산한다

2 부분 문제의 인수를 매번 무작위로 바꾼다

3 같은 부분 문제의 결과를 저장하여 재사용한다

4 재귀 호출 깊이만 줄이면 모든 중복이 사라진다

5 기저 사례의 값을 항상 하나로 통일한다

문제 02 정답과 해설

동적 계획법이 중복 계산을 줄이는 방식은?

3번 같은 부분 문제의 결과를 저장하여 재사용한다

해설

메모이제이션과 표 채우기는 모두 부분 문제의 답을 재사용한다.

문제 03

최적 부분 구조의 의미는?

1. 모든 국소 최적 선택이 전체 최적해로 이어진다
2. 최적해를 구성하는 관련 부분해도 그 부분 문제에 대해 최적이다
3. 관련 부분 문제의 크기는 반드시 절반이다
4. 모든 최적해가 같은 선택 순서를 갖는다
5. 최적 부분 구조가 있으면 중복 부분 문제도 반드시 있다

문제 03 정답과 해설

최적 부분 구조의 의미는?

2번 최적해를 구성하는 관련 부분해도 그 부분 문제에 대해 최적이다

해설

부분해를 더 좋은 해로 대체할 수 있다면 원래 해의 최적성과 모순이 되는 구조를 말한다.

문제 04

하향식 메모이제이션의 일반적인 특성은?

1. 결과를 저장해도 같은 상태를 매번 재계산한다
2. 모든 상태를 반드시 계산한 뒤 재귀를 시작한다
3. 재귀 깊이가 입력 크기와 무관하게 항상 일정하다
4. 재귀 중 필요한 상태를 계산하고 결과를 저장한다
5. 호출 스택을 전혀 사용하지 않는다

문제 04 정답과 해설

하향식 메모이제이션의 일반적인 특성은?

4번 재귀 중 필요한 상태를 계산하고 결과를 저장한다

해설

필요한 상태만 방문할 수 있지만 재귀 호출 스택 비용도 고려해야 한다.

문제 05

상향식 DP의 계산 순서를 정하는 기준은?

1 각 상태가 참조하는 선행 상태를 먼저 계산한다

2 최종 상태부터 의존성을 무시하고 계산한다

3 초기값이 0이면 어느 순서로 계산해도 같다

4 가장 큰 후보값이 있는 상태부터 확정한다

5 모든 상태를 숫자 크기의 역순으로 계산한다

문제 05 정답과 해설

상향식 DP의 계산 순서를 정하는 기준은?

1번 각 상태가 참조하는 선행 상태를 먼저 계산한다

해설

의존 관계를 만족하는 순서여야 아직 계산하지 않은 값을 읽지 않는다.

문제 06

DP 값 0이 유효한 답일 수 있다. memo[x]==0을 미계산 판정으로 쓰면 생길 수 있는 문제는?

1. 정답이 0인 상태가 이미 계산된 것으로 항상 구별된다
2. 정답이 0인 상태를 반복해서 계산할 수 있다
3. 0인 상태의 값을 자동으로 −1로 바꾼다
4. 배열을 0으로 초기화하면 모든 점화식이 정확해진다
5. 캐시가 있으면 재계산 여부와 무관하게 선형 시간이 된다

문제 06 정답과 해설

DP 값 0이 유효한 답일 수 있다. memo[x]==0을 미계산 판정으로 쓰면 생길 수 있는 문제는?

2번 정답이 0인 상태를 반복해서 계산할 수 있다

해설

미계산 여부는 별도 방문 배열이나 답의 범위 밖 표식으로 구분하는 편이 정확하다.

문제 07

F(0)=0, F(1)=1인 피보나치 수의 점화식은?

1 $F(n)=2F(n-1),\ n\geq 2$

2 $F(n)=F(n-1)-F(n-2)$

3 $F(n)=F(n/2)$, 항상

4 $F(n)=n^2$

5 $F(n)=F(n-1)+F(n-2),\ n\geq 2$

문제 07 정답과 해설

F(0)=0, F(1)=1인 피보나치 수의 점화식은?

5번 $F(n)=F(n-1)+F(n-2),\ n\geq 2$

해설

마지막 두 값을 더해 다음 값을 구한다.

문제 08

반복 피보나치에서 다음 항 하나를 a+b로 계산한다. 덧셈 한 번을 기본 연산으로 셀 때, 한 단계의 시간 복잡도는?

1. Θ(2^n)
2. Θ(n²)
3. Θ(n)
4. Θ(n log n)
5. Θ(1)

문제 08 정답과 해설

반복 피보나치에서 다음 항 하나를 a+b로 계산한다. 덧셈 한 번을 기본 연산으로 셀 때, 한 단계의 시간 복잡도는?

5번 Θ(1)

해설

한 단계에서 덧셈과 정해진 수의 대입만 하므로 Θ(1)이다. 전체 n단계의 시간과 구별한다.

문제 09

2×n 보드를 1×2 도미노만으로 빈틈없이 채운다. D(0)의 올바른 값은?

1 2

2 0

3 −1

4 정의할 수 없음

5 1

문제 09 정답과 해설

2×n 보드를 1×2 도미노만으로 빈틈없이 채운다. D(0)의 올바른 값은?

5번 1

해설

빈 보드를 채우는 방법을 한 가지로 두면 마지막 두 칸을 가로 도미노 두 개로 채우는 경우를 일관되게 셀 수 있다.

문제 10

2×n 도미노 타일링에서 마지막 열을 덮는 경우를 나누면 얻는 점화식은?

1 $D(n)=nD(n-1)$

2 $D(n)=D(n-1)^2$

3 $D(n)=D(n/2)$

4 $D(n)=D(n-1)+D(n-2)$

5 $D(n)=2D(n-1)+D(n-2)$

문제 10 정답과 해설

2×n 도미노 타일링에서 마지막 열을 덮는 경우를 나누면 얻는 점화식은?

4번 $D(n)=D(n-1)+D(n-2)$

해설

세로 도미노 하나를 놓는 경우와 가로 도미노 두 개를 놓는 경우로 나뉜다.

문제 11

정수 x를 1로 만들 때 −1, 2·3·5로 나누기(나누어떨어질 때만)를 허용한다.
dp[1]은?

1 −1

2 ∞

3 1

4 2

5 0

문제 11 정답과 해설

정수 x를 1로 만들 때 −1, 2·3·5로 나누기(나누어떨어질 때만)를 허용한다. dp[1]은?

5번 0

해설

이미 목표인 1이므로 연산이 필요 없다.

문제 12

dp[x]는 정수 x를 1로 만드는 최소 연산 수다. 허용 연산은 −1, 나누어떨어질 때 ÷2·÷3·÷5이다. x≥2에서 항상 가능한 −1 연산을 사용한 후보는?

1 dp[x/2]+1, 홀수에서도

2 dp[x−1]+1

3 dp[x−1]

4 dp[x/5]+1, 나누어떨어짐과 무관

5 dp[x−1]−1

문제 12 정답과 해설

dp[x]는 정수 x를 1로 만드는 최소 연산 수다. 허용 연산은 −1, 나누어떨어질 때 ÷2·÷3·÷5이다. x≥2에서 항상 가능한 −1 연산을 사용한 후보는?

2번 dp[x−1]+1

해설

x에서 x−1로 가는 한 번의 연산과, x−1에서 1로 가는 최소 횟수 dp[x−1]을 더한다.

문제 13

x가 3으로 나누어떨어질 때 1로 만들기 DP에 추가하는 후보는?

1 dp[3]+x

2 3×dp[x/3]

3 dp[x/3]+1

4 dp[x/3]−1

5 dp[x−3]

문제 13 정답과 해설

x가 3으로 나누어떨어질 때 1로 만들기 DP에 추가하는 후보는?

3번 dp[x/3]+1

해설

한 번 나눈 뒤 남은 최적 연산 수를 더한다.

문제 14

한 번에 1칸 또는 2칸 올라가는 계단에서 순서를 구분한 방법 수 S(n)의 점화식은?

1. $S(n)=S(n-1)\times S(n-2)$
2. $S(n)=S(n/2)$
3. $S(n)=2n$
4. $S(n)=n!$
5. $S(n)=S(n-1)+S(n-2)$

문제 14 정답과 해설

한 번에 1칸 또는 2칸 올라가는 계단에서 순서를 구분한 방법 수 S(n)의 점화식은?

5번 $S(n)=S(n-1)+S(n-2)$

해설

마지막 이동이 1칸인지 2칸인지로 나누면 겹치지 않는 두 경우가 된다.

문제 15

0/1 배낭에서 dp[i][c]를 앞 i개 물건으로 용량 c 이하에서 얻는 최대 가치로 정의한다. 물건을 안 담는 후보는?

1. dp[i−1][c−w_i]+w_i
2. v_i, 항상
3. dp[i][c]+v_i
4. dp[i−1][c]
5. dp[i][c−1], 반드시

문제 15 정답과 해설

0/1 배낭에서 dp[i][c]를 앞 i개 물건으로 용량 c 이하에서 얻는 최대 가치로 정의한다. 물건을 안 담는 후보는?

4번 dp[i−1][c]

해설

i번째 물건을 제외하면 앞 i−1개로 같은 용량을 쓰는 문제다.

문제 16

0/1 배낭에서 $w_i \le c$일 때 i번째 물건을 담는 후보는?

1. $dp[i-1][c-w_i]-v_i$
2. $dp[i-1][c-w_i]+v_i$
3. $dp[i][c-w_i]+v_i$
4. $dp[i-1][c]+v_i$
5. $dp[i-1][c-v_i]+w_i$

문제 16 정답과 해설

0/1 배낭에서 $w_i \le c$일 때 i번째 물건을 담는 후보는?

2번 $dp[i-1][c-w_i]+v_i$

해설

이전 행을 참조해야 같은 물건을 다시 담지 않는다.

문제 17

0/1 배낭을 1차원 dp[c]로 압축할 때 물건별 용량 순회 방향은?

1. 용량이 물건 무게인 칸만 갱신한다
2. 물건별로 같은 방향을 정하지 않고 번갈아 순회한다
3. 용량별 후보를 계산하지 않고 최대 가치 물건만 남긴다
4. 큰 용량에서 작은 용량으로
5. 작은 용량부터 현재 dp 값을 바로 재사용한다

문제 17 정답과 해설

0/1 배낭을 1차원 dp[c]로 압축할 때 물건별 용량 순회 방향은?

4번 큰 용량에서 작은 용량으로

해설

큰 용량부터 갱신하면 dp[c−w]는 현재 물건을 쓰기 전의 값으로 남는다.

문제 18

물건을 무제한 재사용하는 배낭에서 현재 물건을 여러 번 쓸 수 있도록 하는 1차원 순회 방향은?

1. 용량별 갱신을 한 번도 하지 않고 최대 가치만 비교
2. 용량이 물건 무게와 같은 칸만 갱신
3. 작은 용량에서 큰 용량으로
4. 큰 용량에서 작은 용량으로
5. 한 물건에서 용량마다 이전 물건 단계만 참조

문제 18 정답과 해설

물건을 무제한 재사용하는 배낭에서 현재 물건을 여러 번 쓸 수 있도록 하는 1차원 순회 방향은?

3번 작은 용량에서 큰 용량으로

해설

앞서 갱신된 현재 물건의 상태를 다시 참조해야 반복 사용이 반영된다.

문제 19

용량 이하 배낭에서 물건을 하나도 고르지 않아도 된다. dp[0][c]는?

1 c 자체

2 모든 c에서 ∞

3 −c

4 모든 c≥0에서 0

5 물건 수

문제 19 정답과 해설

용량 이하 배낭에서 물건을 하나도 고르지 않아도 된다. dp[0][c]는?

4번 모든 c≥0에서 0

해설

선택 가능한 물건이 없으므로 얻을 수 있는 가치는 0이다.

문제 20

0/1 배낭 DP가 물건 i=1..n마다 용량 w=0..W를 한 번씩 확인한다. 한 칸 계산은 O(1)이다. W ≥1일 때 총 시간 복잡도는?

1 Θ(n)

2 Θ(W)

3 Θ(log W)

4 Θ(n+W)

5 Θ(nW)

문제 20 정답과 해설

0/1 배낭 DP가 물건 i=1..n마다 용량 w=0..W를 한 번씩 확인한다. 한 칸 계산은 O(1)이다. W≥1일 때 총 시간 복잡도는?

5번 Θ(nW)

해설

n개의 물건마다 W+1칸을 계산한다. n(W+1)은 W≥1에서 Θ(nW)이다.

문제 21

부분 수열과 부분 문자열의 차이는?

1. 부분 수열은 중복 문자를 새로 복제할 수 있다
2. 부분 수열은 선택한 문자의 순서를 자유롭게 바꾼다
3. 부분 문자열은 연속일 필요가 없다
4. 부분 수열은 원래 순서를 유지하지만 연속일 필요는 없다
5. 부분 수열과 부분 문자열은 최대 길이가 항상 같다

문제 21 정답과 해설

부분 수열과 부분 문자열의 차이는?

4번 부분 수열은 원래 순서를 유지하지만 연속일 필요는 없다

해설

LCS는 떨어진 문자들을 선택할 수 있지만 순서를 뒤집을 수는 없다.

문제 22

LCS의 dp[i][j]를 두 문자열의 길이 i,j 접두사의 LCS 길이로 정의한다. 끝 문자가 같으면?

1. 0, 항상
2. dp[i−1][j]+1, 항상
3. dp[i−1][j−1]+1
4. dp[i][j−1]+1, 항상
5. dp[i−1][j−1]−1

문제 22 정답과 해설

LCS의 dp[i][j]를 두 문자열의 길이 i,j 접두사의 LCS 길이로 정의한다. 끝 문자가 같으면?

3번 dp[i−1][j−1]+1

해설

같은 끝 문자를 공통 부분 수열의 마지막 문자로 붙일 수 있다.

문제 23

LCS에서 두 접두사의 끝 문자가 다르면?

1 min(dp[i−1][j],dp[i][j−1])−1

2 dp[i−1][j]+dp[i][j−1]

3 dp[i−1][j−1]+1

4 1, 항상

5 max(dp[i−1][j],dp[i][j−1])

문제 23 정답과 해설

LCS에서 두 접두사의 끝 문자가 다르면?

5번 max(dp[i−1][j],dp[i][j−1])

해설

둘 중 한쪽 끝 문자를 제외하는 경우를 비교한다.

문제 24

LCS의 경계값 dp[0][j], dp[i][0]은?

1 각각 j와 i

2 모두 0

3 모두 1

4 모두 ∞

5 모두 −1

문제 24 정답과 해설

LCS의 경계값 dp[0][j], dp[i][0]은?

2번 모두 0

해설

한 문자열이 비어 있으면 공통 부분 수열의 길이는 0이다.

문제 25

길이 m,n인 두 문자열의 표준 2차원 LCS DP 시간은?

1 Θ(mn)

2 Θ(1)

3 Θ(log mn)

4 Θ(m+n), 항상

5 Θ(m!n!)

문제 25 정답과 해설

길이 m,n인 두 문자열의 표준 2차원 LCS DP 시간은?

1번 Θ(mn)

해설

각 접두사 쌍의 상태를 한 번씩 상수 시간에 계산한다.

문제 26

LCS 길이 표에서 현재 칸을 계산할 때 참조하는 값에 대한 올바른 설명은?

1. 현재 행은 이전 행과 현재 행의 바로 왼쪽 값만 참조한다
2. 위와 왼쪽 값이 언제나 같으므로 하나를 생략할 수 있다
3. 문자가 다를 때에도 항상 왼쪽 위 값에 1을 더한다
4. 현재 행은 왼쪽 위 값만 참조하므로 한 변수면 충분하다
5. 이전 행을 지워도 현재 행의 모든 값이 독립적으로 계산된다

문제 26 정답과 해설

LCS 길이 표에서 현재 칸을 계산할 때 참조하는 값에 대한 올바른 설명은?

1번 현재 행은 이전 행과 현재 행의 바로 왼쪽 값만 참조한다

해설

문자가 같으면 왼쪽 위 값에 1을 더하고, 다르면 위와 왼쪽 값 중 큰 값을 사용한다.

문제 27

LCS 복원에서 두 문자가 같아 답에 문자를 추가한 뒤 이동하는 방향은?

1. i 그대로,j−1
2. i−1,j 그대로
3. i+1,j+1
4. 항상 첫 칸으로
5. i−1,j−1

문제 27 정답과 해설

LCS 복원에서 두 문자가 같아 답에 문자를 추가한 뒤 이동하는 방향은?

5번 i−1,j−1

해설

끝 문자를 사용했으므로 두 문자열의 그 문자를 모두 제외한 상태로 간다.

문제 28

동일한 최장 길이를 갖는 공통 부분 수열이 여러 개일 때 옳은 것은?

1. 복원 동률 규칙에 따라 서로 다른 정답 문자열이 나올 수 있다
2. 동률에서는 두 문자를 모두 붙여도 최장 길이가 유지된다
3. 서로 다른 수열이 나오면 둘 중 하나는 반드시 오답이다
4. 항상 사전순으로 가장 작은 문자열이 복원된다
5. 최적 길이도 복원 규칙에 따라 달라진다

문제 28 정답과 해설

동일한 최장 길이를 갖는 공통 부분 수열이 여러 개일 때 옳은 것은?

1번 복원 동률 규칙에 따라 서로 다른 정답 문자열이 나올 수 있다

해설

정답을 하나만 요구하면 길이가 최대이고 양쪽의 부분 수열인지 확인하면 된다.

문제 29

인접 창고를 동시에 고르지 않는 최대 합 문제에서 D[i]를 앞 i개 창고의 최댓값으로 정의한다. i≥2일 때 점화식은?

1 $\min(D[i-1], D[i-2])$

2 $D[i-1] \times a[i-1]$

3 $D[i-2] - a[i-1]$

4 $\max(D[i-1], D[i-2]+a[i-1])$

5 $D[i-1]+a[i-1]$, 항상

문제 29 정답과 해설

인접 창고를 동시에 고르지 않는 최대 합 문제에서 D[i]를 앞 i개 창고의 최댓값으로 정의한다. i≥2일 때 점화식은?

4번 max(D[i−1],D[i−2]+a[i−1])

해설

마지막 창고를 안 고르거나, 고르고 바로 앞 창고를 제외한다.

문제 30

인접 선택 금지 문제에서 음수 값이 있고 아무것도 고르지 않아도 된다. 창고 하나의 초기값은?

1. -1
2. max(0,a[0])
3. a[0], 무조건
4. ∞
5. 1

문제 30 정답과 해설

인접 선택 금지 문제에서 음수 값이 있고 아무것도 고르지 않아도 된다. 창고 하나의 초기값은?

2번 max(0,a[0])

해설

음수 창고 하나보다 빈 선택의 합 0이 더 좋을 수 있다.

문제 31

오른쪽·아래로만 이동하는 격자 최소 비용 DP에서 일반 칸의 선행 상태는?

1 위 칸과 왼쪽 칸

2 상하좌우 전부

3 아래 칸과 오른쪽 칸

4 시작 칸만

5 대각선 아래 칸만

문제 31 정답과 해설

오른쪽·아래로만 이동하는 격자 최소 비용 DP에서 일반 칸의 선행 상태는?

1번 위 칸과 왼쪽 칸

해설

현재 칸에 들어올 수 있는 마지막 이동을 역으로 나열한다.

문제 32

격자 비용에 시작 칸의 비용도 포함한다면 dp[0][0]은?

1. cost[0][0]
2. ∞, 다른 상태가 나중에 갱신하게 함
3. 첫 행의 모든 비용 합
4. 0, 시작 비용과 무관하게
5. 도착 칸의 비용

문제 32 정답과 해설

격자 비용에 시작 칸의 비용도 포함한다면 dp[0][0]은?

1번 cost[0][0]

해설

시작 상태의 비용 포함 여부를 문제 명세와 맞춰야 한다.

문제 33

오른쪽·아래 이동으로 도착하면 칸 비용 c를, 오른쪽 아래 대각선으로 도착하면 floor(c/2)를 낸다. 대각선 후보는?

1. dp[i−1][j−1]+floor(cost[i][j]/2)
2. dp[i−1][j−1]−cost[i][j]
3. dp[i−1][j−1]/2+cost[i][j]
4. cost[i−1][j−1]/2
5. floor(dp[i][j]/2)

문제 33 정답과 해설

오른쪽·아래 이동으로 도착하면 칸 비용 c를, 오른쪽 아래 대각선으로 도착하면 floor(c/2)를 낸다.
대각선 후보는?

1번 dp[i−1][j−1]+floor(cost[i][j]/2)

해설

할인은 전체 경로 비용이 아니라 이번에 들어가는 칸의 비용에만 적용된다.

문제 34

격자의 왼쪽 위에서 시작해 오른쪽·아래·오른쪽 아래 대각선으로만 이동한다. 격자 밖으로 나갈 수 없다. 첫 행의 시작 칸 오른쪽에 있는 칸으로 들어올 수 있는 방향은?

1. 격자 밖 칸을 비용 0으로 두고 세 방향 모두
2. 왼쪽에서 오는 오른쪽 이동
3. 왼쪽 이동과 대각선 이동 모두
4. 왼쪽 위에서 오는 대각선 이동
5. 위에서 오는 아래 이동

문제 34 정답과 해설

격자의 왼쪽 위에서 시작해 오른쪽·아래·오른쪽 아래 대각선으로만 이동한다. 격자 밖으로 나갈 수 없다. 첫 행의 시작 칸 오른쪽에 있는 칸으로 들어올 수 있는 방향은?

2번 왼쪽에서 오는 오른쪽 이동

해설

첫 행의 위나 왼쪽 위는 격자 밖이다. 따라서 왼쪽 칸에서 오른쪽으로 오는 이동만 가능하다.

문제 35

순환 의존성이 없는 DP에서 계산 순서를 그래프 관점으로 표현하면?

1 상태 의존 그래프의 위상 순서

2 상태 값을 큰 순서로 확정하는 순서

3 인덱스 역순, 의존성과 무관하게

4 모든 경계 상태를 마지막에 계산하는 순서

5 그래프의 간선을 가중치순으로 정렬한 순서

문제 35 정답과 해설

순환 의존성이 없는 DP에서 계산 순서를 그래프 관점으로 표현하면?

1번 상태 의존 그래프의 위상 순서

해설

각 상태의 선행 상태가 먼저 계산되는 순서다.

문제 36

일반 TSP를 현재 도시 하나만으로 DP 상태화하면 부족한 이유는?

1. 시작 도시만 기억하면 나머지 방문 여부를 추론할 수 있다
2. 완전 그래프에서는 모든 방문 집합이 같은 부분 문제다
3. 방문한 도시 집합에 따라 앞으로 갈 수 있는 도시가 달라진다
4. 현재 도시에 도착한 비용이 같으면 방문 이력은 항상 무관하다
5. 방문한 도시 수만 추가하면 방문 집합을 항상 대체할 수 있다

문제 36 정답과 해설

일반 TSP를 현재 도시 하나만으로 DP 상태화하면 부족한 이유는?

3번 방문한 도시 집합에 따라 앞으로 갈 수 있는 도시가 달라진다

해설

현재 도시가 같아도 방문 이력이 다르면 같은 부분 문제로 합칠 수 없다.

문제 37

DP의 시간 복잡도를 계산하는 기본 방법은?

1. 전이 후보 수만 세고 상태 수는 제외한다
2. 상태 수만 세고 각 상태의 반복은 제외한다
3. 상태 수와 상태당 전이 비용을 곱한다
4. 재귀 호출 깊이만으로 전체 시간을 정한다
5. 메모이제이션이면 상태 개수와 무관하게 O(n)이다

문제 37 정답과 해설

DP의 시간 복잡도를 계산하는 기본 방법은?

3번 상태 수와 상태당 전이 비용을 곱한다

해설

각 상태가 한 번 계산되는지 확인하고 전이의 반복 비용을 포함한다.

문제 38

최솟값 DP에서 불가능 상태를 0으로 초기화하면 생길 수 있는 오류는?

1. 경계 상태를 모두 0으로 두면 입력 비용과 무관하게 정확하다
2. 불가능 상태의 0은 최솟값 비교에서 자동으로 제외된다
3. 존재하지 않는 경로가 비용 0의 유리한 후보로 선택된다
4. 불가능 상태의 값은 이후 전이에 영향을 주지 않는다
5. 큰 양수로 초기화하면 오히려 없는 경로가 선택된다

문제 38 정답과 해설

최솟값 DP에서 불가능 상태를 0으로 초기화하면 생길 수 있는 오류는?

3번 존재하지 않는 경로가 비용 0의 유리한 후보로 선택된다

해설

불가능 상태는 ∞ 같은 표식으로 두고 유효한 기저 상태만 실제 값으로 설정한다.

문제 39

DP에서 최적값뿐 아니라 실제 선택도 복원하려면 일반적으로 필요한 정보는?

1. 최종 최적값 하나만 있으면 모든 선택을 항상 유일하게 복원한다
2. 입력을 정렬한 순서만 저장하면 충분하다
3. 모든 상태의 최적값이 같다고 가정하고 역순으로 출력한다
4. 어느 선행 상태나 선택으로 최적값을 얻었는지에 대한 정보
5. 최적값과 같은 횟수만큼 첫 선택을 반복한다

문제 39 정답과 해설

DP에서 최적값뿐 아니라 실제 선택도 복원하려면 일반적으로 필요한 정보는?

4번 어느 선행 상태나 선택으로 최적값을 얻었는지에 대한 정보

해설

선택 정보를 저장하거나 완성된 표에서 같은 전이를 역추적할 수 있다.

문제 40

동적 계획법과 탐욕법을 구별하는 설명으로 적절한 것은?

1 재귀 함수이면 DP이고 반복문이면 탐욕법이다

2 DP는 정의한 상태의 여러 전이 후보를 비교하고 결과를 재사용한다

3 최적 부분 구조가 있으면 DP와 탐욕법이 항상 같은 선택을 한다

4 배열을 사용하면 문제 구조와 무관하게 DP다

5 탐욕법도 반드시 모든 부분 문제의 값을 저장한다

문제 40 정답과 해설

동적 계획법과 탐욕법을 구별하는 설명으로 적절한 것은?

2번 DP는 정의한 상태의 여러 전이 후보를 비교하고 결과를 재사용한다

해설

문법이나 재귀 유무가 아니라 문제를 해결하는 선택 및 재사용 구조로 구분한다.

문제 41

F(0)=0,F(1)=1일 때 F(2)부터 F(7)까지 쓰시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 41 정답과 해설

F(0)=0,F(1)=1일 때 F(2)부터 F(7)까지 쓰시오.

1,2,3,5,8,13

해설

앞의 두 값을 더해 순서대로 계산한다.

문제 42

F(0),F(1)이 미리 저장되어 있다. 메모이제이션으로 F(6)을 처음 구할 때 새로 계산하여 저장하는 비기저 상태의 수는?

풀이 과정과 판단 근거를 함께 쓰시오.

문제 42 정답과 해설

F(0),F(1)이 미리 저장되어 있다. 메모이제이션으로 F(6)을 처음 구할 때 새로 계산하여 저장하는 비기저 상태의 수는?

5개

해설

F(2),F(3),F(4),F(5),F(6)을 각각 한 번 계산한다. 호출 횟수 전체와 구별한다.

문제 43

변수 a=0,b=1에서 (a,b)=(b,a+b)를 5회 동시에 갱신한다. 최종 값은?

풀이 과정과 판단 근거를 함께 쓰시오.

문제 43 정답과 해설

변수 a=0,b=1에서 (a,b)=(b,a+b)를 5회 동시에 갱신한다. 최종 값은?

(a,b)=(5,8)

해설

각 단계에서 a는 F(k), b는 F(k+1)이 된다.

문제 44

2×n 도미노 타일링에서 D(0)=D(1)=1이다. D(2)부터 D(5)를 쓰시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 44 정답과 해설

2×n 도미노 타일링에서 D(0)=D(1)=1이다. D(2)부터 D(5)를 쓰시오.

2,3,5,8

해설

각 항은 앞의 두 항의 합이다.

문제 45

한 번에 1칸 또는 2칸 오를 때 4칸을 오르는 방법을 모두 쓰시오. 이동 순서를 구분한다.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 45 정답과 해설

한 번에 1칸 또는 2칸 오를 때 4칸을 오르는 방법을 모두 쓰시오. 이동 순서를 구분한다.

1111,112,121,211,22의 5가지

해설

마지막 이동을 기준으로 3칸의 3가지와 2칸의 2가지를 합해도 5다.

문제 46

2×6 도미노 타일링 수를 5로 나눈 나머지는?

풀이 과정과 판단 근거를 함께 쓰시오.

문제 46 정답과 해설

2×6 도미노 타일링 수를 5로 나눈 나머지는?

3

해설

D(6)=13이므로 13 mod 5=3이다.

문제 47

−1, ÷2, ÷3, ÷5를 허용하는 1로 만들기에서 dp[1]부터 dp[6]을 쓰시오. 나눗셈은 나누어떨어질 때만 가능하다.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 47 정답과 해설

−1, ÷2, ÷3, ÷5를 허용하는 1로 만들기에서 dp[1]부터 dp[6]을 쓰시오. 나눗셈은 나누어떨어질 때만 가능하다.

[0,1,1,2,1,2]

해설

4는 2를 거쳐 두 번, 6은 3 또는 2를 거쳐 두 번에 도달한다.

문제 48

−1, ÷2, ÷3, ÷5로 26을 1로 만드는 최소 연산 수와 경로 하나를 쓰시오. 나눗셈은 나누어떨어질 때만 가능하다.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 48 정답과 해설

−1, ÷2, ÷3, ÷5로 26을 1로 만드는 최소 연산 수와 경로 하나를 쓰시오. 나눗셈은 나누어떨어질 때만 가능하다.

3회, 26→25→5→1

해설

26에서 먼저 2로 나누는 것보다 1을 빼고 5로 나누는 경로가 짧다. 2회 이내 도달은 불가능하다.

문제 49

−1, ÷2, ÷3, ÷5를 허용한다. 10에서 가능한 한 번의 다음 값과 최소 연산 수를 쓰시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 49 정답과 해설

−1, ÷2, ÷3, ÷5를 허용한다. 10에서 가능한 한 번의 다음 값과 최소 연산 수를 쓰시오.

다음 값 {9,5,2}, 최소 2회

해설

10→5→1 또는 10→2→1이다. 10은 3으로 나누어떨어지지 않는다.

문제 50

−1, ÷2, ÷3, ÷5로 7을 1로 만드는 최소 연산 수와 경로 하나를 쓰시오. 나눗셈은 나누어떨어질 때만 가능하다.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 50 정답과 해설

−1, ÷2, ÷3, ÷5로 7을 1로 만드는 최소 연산 수와 경로 하나를 쓰시오. 나눗셈은 나누어떨어질 때만 가능하다.

3회, 7→6→3→1

해설

첫 연산은 −1만 가능하다. 6에서 두 번 더 필요하다.

문제 51

0/1 배낭의 물건 (무게,가치)가 (2,3),(3,4), 용량 5다. 첫 물건 처리 후 용량 0..5의 행은?

풀이 과정과 판단 근거를 함께 쓰시오.

문제 51 정답과 해설

0/1 배낭의 물건 (무게,가치)가 (2,3),(3,4), 용량 5다. 첫 물건 처리 후 용량 0..5의 행은?

[0,0,3,3,3,3]

해설

첫 물건은 최대 한 번만 사용한다. 용량 4여도 가치 6을 만들 수 없다.

문제 52

0/1 배낭의 물건 (2,3),(3,4), 용량 5다. 두 번째 물건까지 처리한 용량 0..5의 행은?

풀이 과정과 판단 근거를 함께 쓰시오.

문제 52 정답과 해설

0/1 배낭의 물건 (2,3),(3,4), 용량 5다. 두 번째 물건까지 처리한 용량 0..5의 행은?

[0,0,3,4,4,7]

해설

용량 3과 4는 두 번째만 담고, 용량 5는 둘 다 담는다.

문제 53

용량 7, 0/1 물건 A(3,4), B(4,5), C(2,3)의 최적 구성과 가치는?

풀이 과정과 판단 근거를 함께 쓰시오.

문제 53 정답과 해설

용량 7, 0/1 물건 A(3,4), B(4,5), C(2,3)의 최적 구성과 가치는?

A+B, 가치 9

해설

A+C는 7, B+C는 8이다. 세 물건의 무게 합은 9라서 불가능하다.

문제 54

0/1 물건 한 개 (무게 2,가치 3), 용량 4다. 1차원 DP를 잘못 오름차순 갱신하면 dp[4]는 얼마가 되고 실제 답은 얼마인가?

풀이 과정과 판단 근거를 함께 쓰시오.

문제 54 정답과 해설

0/1 물건 한 개 (무게 2,가치 3), 용량 4다. 1차원 DP를 잘못 오름차순 갱신하면 dp[4]는 얼마가 되고 실제 답은 얼마인가?

잘못된 값 6, 실제 답 3

해설

dp[2]=3을 만든 뒤 같은 물건으로 dp[4]=dp[2]+3을 계산하여 중복 사용한다.

문제 55

무제한 사용 가능한 물건 한 종류 (무게 2,가치 3), 용량 5의 최대 가치는?

풀이 과정과 판단 근거를 함께 쓰시오.

문제 55 정답과 해설

무제한 사용 가능한 물건 한 종류 (무게 2,가치 3), 용량 5의 최대 가치는?

6

해설

두 개를 담아 무게 4, 가치 6을 얻는다. 남은 용량 1은 비워도 된다.

문제 56

용량 3, 0/1 물건 (무게 4,가치 100),(무게 2,가치 5)의 최대 가치는?

풀이 과정과 판단 근거를 함께 쓰시오.

문제 56 정답과 해설

용량 3, 0/1 물건 (무게 4,가치 100),(무게 2,가치 5)의 최대 가치는?

5

해설

첫 물건은 들어가지 않는다. 둘째 물건 하나를 선택한다.

문제 57

문자열 ABC와 AC의 LCS 길이와 수열을 쓰시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 57 정답과 해설

문자열 ABC와 AC의 LCS 길이와 수열을 쓰시오.

길이 2, AC

해설

B를 건너뛰면 AC를 얻는다. 두 번째 문자열 전체를 공통 부분 수열로 쓸 수 있다.

문제 58

문자열 AB와 BA의 LCS 길이와 가능한 모든 최장 수열을 쓰시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 58 정답과 해설

문자열 AB와 BA의 LCS 길이와 가능한 모든 최장 수열을 쓰시오.

길이 1, A 또는 B

해설

길이 2의 공통 부분 수열은 순서가 맞지 않아 존재하지 않는다.

문제 59

문자열 AAB와 AB의 LCS DP 마지막 행을 쓰시오. 열은 빈 문자열, A, AB 순이다.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 59 정답과 해설

문자열 AAB와 AB의 LCS DP 마지막 행을 쓰시오. 열은 빈 문자열, A, AB 순이다.

[0,1,2]

해설

첫 A 하나만 사용할 때 길이 1, B까지 맞추면 길이 2다.

문제 60

문자열 ABC와 DEF의 LCS 길이는?

풀이 과정과 판단 근거를 함께 쓰시오.

문제 60 정답과 해설

문자열 ABC와 DEF의 LCS 길이는?

0

해설

공통 문자가 하나도 없다.

문제 61

문자열 ABCD와 BACD의 LCS 길이와 수열 하나를 쓰시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 61 정답과 해설

문자열 ABCD와 BACD의 LCS 길이와 수열 하나를 쓰시오.

길이 3, ACD 또는 BCD

해설

A와 B의 상대 순서가 뒤집혀 네 글자 모두 선택할 수 없다. C,D를 뒤에 붙이면 길이 3이다.

문제 62

LCS 표에서 끝 문자가 다르고 위 값 3, 왼쪽 값 4, 왼쪽 위 값 2다. 현재 값은?

풀이 과정과 판단 근거를 함께 쓰시오.

문제 62 정답과 해설

LCS 표에서 끝 문자가 다르고 위 값 3, 왼쪽 값 4, 왼쪽 위 값 2다. 현재 값은?

4

해설

끝 문자가 다르면 위와 왼쪽 중 큰 값만 비교한다.

문제 63

LCS 표에서 끝 문자가 같고 왼쪽 위 값이 3이다. 현재 값은?

풀이 과정과 판단 근거를 함께 쓰시오.

문제 63 정답과 해설

LCS 표에서 끝 문자가 같고 왼쪽 위 값이 3이다. 현재 값은?

4

해설

같은 문자를 공통 부분 수열 끝에 하나 더 붙인다.

문제 64

인접 창고를 동시에 선택하지 않고 빈 선택을 허용한다. 값 [2,7,9,3,1]의 최대 합과 선택 인덱스를 쓰시오. 인덱스는 0부터다.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 64 정답과 해설

인접 창고를 동시에 선택하지 않고 빈 선택을 허용한다. 값 [2,7,9,3,1]의 최대 합과 선택 인덱스를 쓰시오. 인덱스는 0부터다.

합 12, 인덱스 [0,2,4]

해설

2+9+1=12이며 앞 i개 최적값은 0,2,7,11,11,12다.

문제 65

값 [5,1,1,5]의 창고 중 인접한 둘을 함께 고르지 않을 때 최대 합은? 빈 선택도 허용한다.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 65 정답과 해설

값 [5,1,1,5]의 창고 중 인접한 둘을 함께 고르지 않을 때 최대 합은? 빈 선택도 허용한다.

10

해설

첫 창고와 마지막 창고는 인접하지 않으므로 함께 선택할 수 있다.

문제 66

인접 선택을 금지하고 빈 선택을 허용한다. 값 [−4,−2,−7]의 최대 합은?

풀이 과정과 판단 근거를 함께 쓰시오.

문제 66 정답과 해설

인접 선택을 금지하고 빈 선택을 허용한다. 값 [−4,−2,−7]의 최대 합은?

0

해설

음수 창고를 고르지 않는 빈 선택이 최적이다.

문제 67

격자 [[1,4],[2,3]]에서 오른쪽·아래로만 움직인다. 시작과 도착 칸을 포함한 최소 비용은?

풀이 과정과 판단 근거를 함께 쓰시오.

문제 67 정답과 해설

격자 [[1,4],[2,3]]에서 오른쪽·아래로만 움직인다. 시작과 도착 칸을 포함한 최소 비용은?

6

해설

아래로 간 뒤 오른쪽으로 가면 1+2+3=6이다.

문제 68

격자 [[5,9],[8,3]]에서 오른쪽·아래 도착은 칸 비용 전부, 대각선 도착은 칸 비용의 내림 절반을 낸다. 시작 비용은 전부 낸다. 최솟값은?

풀이 과정과 판단 근거를 함께 쓰시오.

문제 68 정답과 해설

격자 [[5,9],[8,3]]에서 오른쪽·아래 도착은 칸 비용 전부, 대각선 도착은 칸 비용의 내림 절반을 낸다. 시작 비용은 전부 낸다. 최솟값은?

6

해설

시작에서 도착으로 대각선 이동하면 5+floor(3/2)=6이다.

문제 69

격자 [[2,8,4],[6,5,3]]이다. 오른쪽·아래 도착은 해당 칸의 비용 전부, 대각선 도착은 내림 절반을 낸다. 시작 칸은 전액을 낸다. 최소 비용 DP의 두 행을 쓰시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 69 정답과 해설

격자 [[2,8,4],[6,5,3]]이다. 오른쪽·아래 도착은 해당 칸의 비용 전부, 대각선 도착은 내림 절반을 낸다. 시작 칸은 전액을 낸다. 최소 비용 DP의 두 행을 쓰시오.

[2,10,14], [8,4,7]

해설

가운데 아래는 2+floor(5/2)=4다. 마지막은 왼쪽에서 4+3=7이 대각선 후보 10+1보다 작다.

문제 70

상태가 (i,c), 0≤i≤n, 0≤c≤W이고 각 상태에서 후보 두 개를 비교한다. 상태 수와 시간 차수를 쓰시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 70 정답과 해설

상태가 (i,c), 0≤i≤n, 0≤c≤W이고 각 상태에서 후보 두 개를 비교한다. 상태 수와 시간 차수를 쓰시오.

(n+1)(W+1)개, O(nW)

해설

경계 상태를 포함한 표 크기는 정확히 곱으로 세며 각 전이는 상수 시간이다.

문제 71

피보나치 단순 재귀에서 중복 부분 문제가 생기는 과정을 F(5)를 이용해 설명하시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 71 정답과 해설

피보나치 단순 재귀에서 중복 부분 문제가 생기는 과정을 F(5)를 이용해 설명하시오.

F(5)는 F(4),F(3)을 호출하고 F(4)도 F(3)을 호출한다.

해설

같은 F(3) 이하의 계산이 여러 경로에서 반복된다. 메모이제이션은 각 인수의 답을 저장하여 재사용한다.

문제 72

메모이제이션의 캐시 키에 필요한 상태 정보를 빠뜨리면 어떤 오류가 생기는지 TSP를 예로 설명하시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 72 정답과 해설

메모이제이션의 캐시 키에 필요한 상태 정보를 빠뜨리면 어떤 오류가 생기는지 TSP를 예로 설명하시오.

현재 도시가 같아도 방문 도시 집합이 다르면 남은 선택이 다르므로 같은 답으로 합치면 안 된다.

해설

키에 현재 도시와 방문 집합을 함께 넣어야 동일한 부분 문제인지 구분할 수 있다.

문제 73

2×n 도미노 타일링 점화식을 마지막 부분의 배치로 증명하시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 73 정답과 해설

2×n 도미노 타일링 점화식을 마지막 부분의 배치로 증명하시오.

오른쪽 끝이 세로 도미노 하나면 2×(n−1), 가로 두 개면 2×(n−2)가 남는다.

해설

두 경우는 겹치지 않고 모든 배치를 포함한다. 따라서 $D(n)=D(n-1)+D(n-2)$, $D(0)=D(1)=1$이다.

문제 74

1로 만들기에서 가능하면 가장 큰 수 5,3,2로 먼저 나누는 탐욕을 26으로 반박하시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 74 정답과 해설

1로 만들기에서 가능하면 가장 큰 수 5,3,2로 먼저 나누는 탐욕을 26으로 반박하시오.

26→13→12→4→2→1은 5회지만 26→25→5→1은 3회다.

해설

26은 처음에 2로만 나눌 수 있다. 첫 선택을 확정하지 말고 모든 합법적인 다음 상태의 최솟값을 비교해야 한다.

문제 75

1로 만들기 DP를 x=2부터 오름차순으로 계산해도 되는 이유와 나눗셈 후보의 조건을 쓰시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 75 정답과 해설

1로 만들기 DP를 x=2부터 오름차순으로 계산해도 되는 이유와 나눗셈 후보의 조건을 쓰시오.

x−1과 x/2,x/3,x/5는 모두 x보다 작다. 각 나눗셈은 해당 수로 나누어떨어질 때만 쓴다.

해설

오름차순이면 선행 상태가 이미 계산되어 있다. 정수 나눗셈으로 소수 부분을 버리는 것은 허용된 연산이 아니다.

문제 76

0/1 배낭의 점화식을 마지막 물건의 선택 여부로 유도하시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 76 정답과 해설

0/1 배낭의 점화식을 마지막 물건의 선택 여부로 유도하시오.

미선택은 dp[i−1][c], 선택 가능하면 dp[i−1][c−w_i]+v_i이며 둘의 최댓값을 취한다.

해설

두 경우가 모든 유효한 해를 나눈다. 무게가 c보다 크면 선택 후보를 만들지 않는다.

문제 77

0/1 배낭의 1차원 갱신을 내림차순으로 해야 하는 이유를 (무게 2,가치 3), 용량 4로 설명하시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 77 정답과 해설

0/1 배낭의 1차원 갱신을 내림차순으로 해야 하는 이유를 (무게 2,가치 3), 용량 4로 설명하시오.

오름차순이면 이번에 만든 dp[2]=3을 읽어 dp[4]=6이 된다. 내림차순은 이전 물건 단계의 dp[2]=0을 읽어 3이 된다.

해설

같은 물건을 두 번 사용하는 것을 막으려면 작은 용량이 아직 이번 물건으로 갱신되지 않아야 한다.

문제 78

정확히 용량 W를 채워야 하는 배낭과 용량 W 이하 배낭의 초기화 차이를 설명하시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 78 정답과 해설

정확히 용량 W를 채워야 하는 배낭과 용량 W 이하 배낭의 초기화 차이를 설명하시오.

정확히 채우기는 dp[0]=0, 양의 무게의 불가능 상태는 −∞로 둔다. 용량 이하는 빈 선택으로 모든 용량에서 0이 가능하다.

해설

정확한 무게 상태를 모두 0으로 두면 존재하지 않는 구성에 가치를 더할 수 있다. 불가능 상태에서는 전이도 주의한다.

문제 79

LCS에서 끝 문자가 다를 때 위와 왼쪽 상태를 비교하는 이유를 쓰시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 79 정답과 해설

LCS에서 끝 문자가 다를 때 위와 왼쪽 상태를 비교하는 이유를 쓰시오.

서로 다른 두 끝 문자를 한 쌍으로 맞출 수 없으므로 적어도 한쪽 끝 문자를 제외해야 한다.

해설

첫 문자열 끝을 제외하거나 둘째 문자열 끝을 제외한 최적 길이의 최댓값을 고른다. 두 값을 더하면 중복 계산한다.

문제 80

문자열 ABC와 AC로 부분 수열과 부분 문자열의 차이를 설명하시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 80 정답과 해설

문자열 ABC와 AC로 부분 수열과 부분 문자열의 차이를 설명하시오.

AC는 ABC에서 B를 건너뛴 부분 수열이지만 연속된 부분 문자열은 아니다.

해설

이 예의 LCS 길이는 2이고 최장 공통 부분 문자열 길이는 1이다. 순서 보존과 연속성은 다른 제약이다.

문제 81

LCS 답안을 문자열 하나로 채점할 때 정답 문자열과 일치하는지만 검사하면 안 되는 이유와 검사 조건을 쓰시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 81 정답과 해설

LCS 답안을 문자열 하나로 채점할 때 정답 문자열과 일치하는지만 검사하면 안 되는 이유와 검사 조건을 쓰시오.

최장 공통 부분 수열이 여러 개일 수 있다. 양쪽 입력의 부분 수열이고 길이가 최적 길이인지 확인한다.

해설

AB와 BA에서는 A와 B가 모두 정답이다. 특정 문자열만 인정하려면 복원 동률 규칙을 문제에 명시해야 한다.

문제 82

LCS 복원 중 답 문자를 뒤에서부터 모았다. 출력 전 필요한 처리와 이유를 설명하시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 82 정답과 해설

LCS 복원 중 답 문자를 뒤에서부터 모았다. 출력 전 필요한 처리와 이유를 설명하시오.

모은 문자 순서를 뒤집는다.

해설

접두사의 끝에서 시작으로 역추적했기 때문에 마지막 문자부터 추가된다. 바로 출력하면 순서가 뒤집힌 수열이 나온다.

문제 83

인접 창고 최대 합의 점화식을 마지막 창고 선택 여부로 설명하시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 83 정답과 해설

인접 창고 최대 합의 점화식을 마지막 창고 선택 여부로 설명하시오.

마지막을 안 고르면 D[i−1], 고르면 D[i−2]+a[i−1]이다.

해설

고른 경우 바로 앞 창고를 제외해야 하므로 두 칸 전 상태를 참조한다. 두 후보의 최댓값이 D[i]다.

문제 84

창고 값에 음수를 허용하면서 빈 선택도 허용한다. D[1]=a[0]이 잘못될 수 있는 예와 수정안을 쓰시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 84 정답과 해설

창고 값에 음수를 허용하면서 빈 선택도 허용한다. D[1]=a[0]이 잘못될 수 있는 예와 수정안을 쓰시오.

입력이 [−5]면 실제 답은 0인데 −5가 된다. D[1]=max(0,a[0])으로 수정한다.

해설

D[0]=0을 두고 점화식에서도 빈 선택을 유지하면 모든 값이 음수인 경우 0을 반환한다.

문제 85

오른쪽·아래 격자 최소 비용을 행 우선 순서로 계산할 수 있는 이유를 쓰시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 85 정답과 해설

오른쪽·아래 격자 최소 비용을 행 우선 순서로 계산할 수 있는 이유를 쓰시오.

각 칸은 위 칸과 왼쪽 칸만 참조하며 두 칸은 행 우선 순서에서 먼저 처리된다.

해설

첫 행과 첫 열은 존재하는 선행 칸만 사용한다. 대각선 이동을 추가해도 왼쪽 위 칸은 이미 처리되어 있다.

문제 86

대각선 이동 시 도착 칸 비용만 절반인 격자에서 (이전 누적비용+현재 비용)/2가 잘못된 이유를 수치 예로 설명하시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 86 정답과 해설

대각선 이동 시 도착 칸 비용만 절반인 격자에서 (이전 누적비용+현재 비용)/2가 잘못된 이유를 수치 예로 설명하시오.

이전 비용 10, 현재 비용 6이면 올바른 값은 10+3=13이지만 잘못된 식은 8이다.

해설

할인은 이번 도착 칸에만 적용되며 이미 지불한 이전 경로 비용은 줄어들지 않는다.

문제 87

격자 최소 비용 DP의 시작 칸을 0으로 놓아도 되는지 비용 정의에 따라 구분하시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 87 정답과 해설

격자 최소 비용 DP의 시작 칸을 0으로 놓아도 되는지 비용 정의에 따라 구분하시오.

시작 칸 비용을 포함하면 cost[0][0]으로, 시작 이후 도착 비용만 세면 0으로 둔다.

해설

점화식이 같아도 초기값의 차이가 모든 경로 비용에 영향을 준다. 문제에 비용 포함 범위를 명시해야 한다.

문제 88

배열 전체를 저장한 DP를 최근 두 값으로 줄일 때 확인해야 할 두 가지를 쓰시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 88 정답과 해설

배열 전체를 저장한 DP를 최근 두 값으로 줄일 때 확인해야 할 두 가지를 쓰시오.

전이가 최근 두 상태만 참조하는지, 실제 해 복원이나 과거 상태 질의가 필요한지 확인한다.

해설

최적값만 구하면 충분해도 복원에는 이전 선택 정보가 필요할 수 있다. 메모리 절약이 요구된 출력을 없애면 안 된다.

문제 89

DP 답이 커져 나머지만 저장하는 방법 수 문제에서 매번 모듈러 연산을 해도 되는 이유와 주의점을 쓰시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 89 정답과 해설

DP 답이 커져 나머지만 저장하는 방법 수 문제에서 매번 모듈러 연산을 해도 되는 이유와 주의점을 쓰시오.

덧셈의 나머지는 각 항의 나머지로 계산해도 같다. 덧셈 자체의 오버플로는 충분한 자료형으로 방지한다.

해설

a+b를 먼저 작은 정수형에서 넘치게 만든 뒤 나머지를 취하면 이미 잘못된 값이다. 최댓값 비교 문제에는 같은 논리를 그대로 적용할 수 없다.

문제 90

DP 풀이를 채점할 때 점화식만 맞고 상태 의미·초기값·계산 순서가 없으면 부족한 이유를 쓰시오.

풀이 과정과 판단 근거를 함께 쓰시오.

문제 90 정답과 해설

DP 풀이를 채점할 때 점화식만 맞고 상태 의미·초기값·계산 순서가 없으면 부족한 이유를 쓰시오.

같은 식도 상태 해석에 따라 답이 달라지고, 잘못된 초기값이나 미계산 상태 참조는 오답을 만든다.

해설

풀이에는 상태의 뜻, 유효 전이, 기저 사례, 의존성을 만족하는 계산 순서와 최종 답의 위치가 필요하다.

문제 91

0≤n≤90. fib_memo는 F(0)=0,F(1)=1을 따른다. memo는 길이 91, 모두 −1로 초기화되어 전달된다. 함수가 방문한 상태를 저장하도록 작성한다. Python 또는 C 중 하나로 함수를 작성하시오.

함수의 동작, 경계 처리, 시간 복잡도를 함께 작성하시오.

문제 91 정답과 해설

메모이제이션 피보나치

이미 계산한 값은 바로 반환하고 미계산 값만 재귀로 구한다.

해설

시간 O(n).

문제 91 Python 구현

메모이제이션 피보나치

```python
def fib_memo(n, memo):
    if memo[n] != -1:
        return memo[n]
    if n < 2:
        memo[n] = n
    else:
        memo[n] = fib_memo(n-1, memo) + fib_memo(n-2, memo)
    return memo[n]
```

시간 O(n).

문제 91 C 구현

메모이제이션 피보나치

```c
long long fib_memo(int n, long long memo[]) {
    if (memo[n]!=-1) return memo[n];
    if (n<2) memo[n]=n;
    else memo[n]=fib_memo(n-1,memo)+fib_memo(n-2,memo);
    return memo[n];
}
```

시간 O(n).

문제 92

2×n 보드를 1×2 도미노로 채우는 수를 m으로 나눈 나머지를 domino(n,m)으로 구한다. $0 \le n \le 10^6$, $1 \le m \le 10^9$. 빈 보드의 방법 수는 1이다. Python 또는 C 중 하나로 함수를 작성하시오.

함수의 동작, 경계 처리, 시간 복잡도를 함께 작성하시오.

문제 92 정답과 해설

도미노 타일링

D(0)=D(1)=1에서 앞 두 값의 합을 반복 계산한다.

해설

시간 O(n).

문제 92 Python 구현

도미노 타일링

```python
def domino(n, m):
    a = b = 1 % m
    for i in range(2, n+1):
        a, b = b, (a+b) % m
    return b
```

시간 O(n).

문제 92 C 구현

도미노 타일링

```c
int domino(int n, int m) {
    long long a=1%m, b=1%m;
    for (int i=2; i<=n; ++i) {
        long long next=(a+b)%m;
        a=b; b=next;
    }
    return (int)b;
}
```

시간 O(n).

문제 93

1≤n≤10000에서 −1, ÷2, ÷3, ÷5로 1에 도달하는 최소 연산 수 make_one을 구한다. 나눗셈은 나누어떨어질 때만 가능하다. C에는 길이 n+1의 dp가 제공된다. Python 또는 C 중 하나로 함수를 작성하시오.

함수의 동작, 경계 처리, 시간 복잡도를 함께 작성하시오.

문제 93 정답과 해설

1로 만들기 최솟값

작은 수부터 모든 합법적인 다음 상태의 최소 연산 수에 1을 더한다.

해설

시간 O(n).

문제 93 Python 구현

1로 만들기 최솟값

```
def make_one(n):
    dp = [0]*(n+1)
    for x in range(2, n+1):
        dp[x] = dp[x-1]+1
        for d in (2,3,5):
            if x % d == 0:
                dp[x] = min(dp[x], dp[x//d]+1)
    return dp[n]
```

시간 O(n).

문제 93 C 구현

1로 만들기 최솟값

```c
int make_one(int n, int dp[]) {
    int divisors[]={2,3,5}; dp[1]=0;
    for (int x=2; x<=n; ++x) {
        dp[x]=dp[x-1]+1;
        for (int j=0; j<3; ++j) {
            int d=divisors[j];
            if (x%d==0 && dp[x/d]+1<dp[x]) dp[x]=dp[x/d]+1;
        }
    }
    return dp[n];
}
```

시간 O(n).

문제 94

1로 만들기의 올바른 dp[1..n]이 제공된다. make_path는 n부터 1까지 최소 경로를 반환한다. 최적 후보 동률은 −1, ÷2, ÷3, ÷5 순이다. C는 out에 쓰고 길이를 반환한다. Python 또는 C 중 하나로 함수를 작성하시오.

함수의 동작, 경계 처리, 시간 복잡도를 함께 작성하시오.

문제 94 정답과 해설

최소 연산 경로 복원

dp가 정확히 1 작은 합법적인 다음 상태 중 명시된 순서의 첫 후보를 선택한다.

해설

1≤n≤10000, out 길이 n이면 충분하다. 경로 길이 L에 대해 시간 O(L), 출력 O(L).

문제 94 Python 구현

최소 연산 경로 복원

```python
def make_path(n, dp):
    path = [n]
    while n > 1:
        candidates = [n-1]
        candidates += [n//d for d in (2,3,5) if n%d == 0]
        n = next(x for x in candidates if dp[x]+1 == dp[n])
        path.append(n)
    return path
```

1≤n≤10000, out 길이 n이면 충분하다. 경로 길이 L에 대해 시간 O(L), 출력 O(L).

문제 94 C 구현

최소 연산 경로 복원

```c
int make_path(int n, const int dp[], int out[]) {
    int length=0, divisors[]={2,3,5};
    out[length++]=n;
    while (n>1) {
        int next=n-1;
        if (dp[next]+1!=dp[n]) {
            for (int j=0; j<3; ++j) {
                int d=divisors[j];
                if (n%d==0 && dp[n/d]+1==dp[n]) {next=n/d; break;}
            }
        }
        n=next; out[length++]=n;
    }
    return length;
}
```

1≤n≤10000, out 길이 n이면 충분하다. 경로 길이 L에 대해 시간 O(L), 출력 O(L).

문제 95

물건 (무게,가치) n≤100개, 용량 W≤1000. 무게 1..1000, 가치 0..10^6. knapsack2는 용량 이하 최대 가치를 반환한다. C dp[101][1001]은 호출자가 제공한다. Python 또는 C 중 하나로 함수를 작성하시오.

함수의 동작, 경계 처리, 시간 복잡도를 함께 작성하시오.

문제 95 정답과 해설

2차원 0/1 배낭

앞 i개 물건을 쓰는 상태에서 선택과 미선택을 이전 행으로 비교한다.

해설

시간 O(nW). 빈 선택을 허용하며 n=0 또는 W=0의 답은 0이다.

문제 95 Python 구현

2차원 0/1 배낭

```python
def knapsack2(items, W):
    dp = [[0]*(W+1) for _ in range(len(items)+1)]
    for i, (w,v) in enumerate(items, 1):
        for c in range(W+1):
            dp[i][c] = dp[i-1][c]
            if w <= c:
                dp[i][c] = max(dp[i][c], dp[i-1][c-w]+v)
    return dp[len(items)][W]
```

시간 O(nW). 빈 선택을 허용하며 n=0 또는 W=0의 답은 0이다.

문제 95 C 구현

2차원 0/1 배낭

```c
int knapsack2(const int w[], const int v[], int n, int W,
              int dp[][1001]) {
    for (int c=0; c<=W; ++c) dp[0][c]=0;
    for (int i=1; i<=n; ++i) {
        for (int c=0; c<=W; ++c) {
            dp[i][c]=dp[i-1][c];
            if (w[i-1]<=c) {
                int take=dp[i-1][c-w[i-1]]+v[i-1];
                if (take>dp[i][c]) dp[i][c]=take;
            }
        }
    }
    return dp[n][W];
}
```

시간 O(nW). 빈 선택을 허용하며 n=0 또는 W=0의 답은 0이다.

문제 96

0/1 물건 n≤100개, 무게 1..1000, 가치 0..10^6, 용량 W=0..1000이다. 길이 W+1 배열로 용량 이하 최대 가치를 구하는 knapsack1을 작성한다. C의 dp는 호출자가 제공한다. Python 또는 C로 작성하시오.

함수의 동작, 경계 처리, 시간 복잡도를 함께 작성하시오.

문제 96 정답과 해설

1차원 0/1 배낭

각 물건에서 용량을 내림차순 갱신하여 같은 물건의 중복 사용을 막는다.

해설

시간 $O(nW)$. 가치 상한은 앞 문항과 같이 10^6이다.

문제 96 Python 구현

1차원 0/1 배낭

```python
def knapsack1(items, W):
    dp = [0]*(W+1)
    for w, v in items:
        for c in range(W, w-1, -1):
            dp[c] = max(dp[c], dp[c-w]+v)
    return dp[W]
```

시간 O(nW). 가치 상한은 앞 문항과 같이 10^6이다.

문제 96 C 구현

1차원 0/1 배낭

```c
int knapsack1(const int w[], const int v[], int n, int W,
              int dp[]) {
    for (int c=0; c<=W; ++c) dp[c]=0;
    for (int i=0; i<n; ++i)
        for (int c=W; c>=w[i]; --c) {
            int take=dp[c-w[i]]+v[i];
            if (take>dp[c]) dp[c]=take;
        }
    return dp[W];
}
```

시간 O(nW). 가치 상한은 앞 문항과 같이 10^6이다.

문제 97

길이 0..100인 ASCII 문자열 a,b의 최장 공통 부분 수열 길이를 lcs_length로 구한다. C는 길이 m,n과 dp[101][101]을 받는다. Python 또는 C 중 하나로 함수를 작성하시오.

함수의 동작, 경계 처리, 시간 복잡도를 함께 작성하시오.

문제 97 정답과 해설

LCS 길이

같으면 왼쪽 위에 1을 더하고, 다르면 위와 왼쪽 중 큰 값을 쓴다.

해설

시간 O(mn). 빈 문자열 경계를 포함한다.

문제 97 Python 구현

LCS 길이

```python
def lcs_length(a, b):
    dp = [[0]*(len(b)+1) for _ in range(len(a)+1)]
    for i in range(1, len(a)+1):
        for j in range(1, len(b)+1):
            if a[i-1] == b[j-1]:
                dp[i][j] = dp[i-1][j-1]+1
            else:
                dp[i][j] = max(dp[i-1][j], dp[i][j-1])
    return dp[len(a)][len(b)]
```

시간 O(mn). 빈 문자열 경계를 포함한다.

문제 97 C 구현

LCS 길이

```c
int lcs_length(const char a[], const char b[], int m, int n,
               int dp[][101]) {
    for (int i=0; i<=m; ++i) dp[i][0]=0;
    for (int j=0; j<=n; ++j) dp[0][j]=0;
    for (int i=1; i<=m; ++i)
        for (int j=1; j<=n; ++j) {
            if (a[i-1]==b[j-1]) dp[i][j]=dp[i-1][j-1]+1;
            else dp[i][j]=dp[i-1][j]>dp[i][j-1]
                          ? dp[i-1][j]:dp[i][j-1];
        }
    return dp[m][n];
}
```

시간 O(mn). 빈 문자열 경계를 포함한다.

문제 98

길이 0..100의 ASCII 문자열 a,b와 접두사 쌍의 LCS 길이 표 dp가 주어진다.
lcs_restore로 수열 하나를 복원한다. 문자 불일치에서 위·왼쪽 동률이면 위로 간다.
C out 길이는 101 이상. Python 또는 C로 작성하시오.

함수의 동작, 경계 처리, 시간 복잡도를 함께 작성하시오.

문제 98 정답과 해설

LCS 수열 복원

끝에서 역추적하며 일치 문자를 모은 뒤 순서를 뒤집는다.

해설

표가 제공되면 복원 시간 O(m+n), 출력 O(min(m,n)).

문제 98 Python 구현

LCS 수열 복원

```python
def lcs_restore(a, b, dp):
    i, j = len(a), len(b)
    out = []
    while i > 0 and j > 0:
        if a[i-1] == b[j-1]:
            out.append(a[i-1]); i -= 1; j -= 1
        elif dp[i-1][j] >= dp[i][j-1]:
            i -= 1
        else:
            j -= 1
    return ''.join(reversed(out))
```

표가 제공되면 복원 시간 O(m+n), 출력 O(min(m,n)).

문제 98 C 구현

LCS 수열 복원

```c
void lcs_restore(const char a[], const char b[], int m, int n,
                 const int dp[][101], char out[]) {
    int i=m, j=n, k=0;
    while (i>0 && j>0) {
        if (a[i-1]==b[j-1]) {out[k++]=a[i-1]; --i; --j;}
        else if (dp[i-1][j]>=dp[i][j-1]) --i;
        else --j;
    }
    out[k]='\0';
    for (i=0; i<k/2; ++i) {
        char t=out[i]; out[i]=out[k-1-i]; out[k-1-i]=t;
    }
}
```

표가 제공되면 복원 시간 O(m+n), 출력 O(min(m,n)).

문제 99

정수 배열 a에서 인접 원소를 함께 고르지 않는 최대 합 nonadjacent를 구한다. 빈 선택 허용, $0 \le n \le 10000$, 값 $-10^6..10^6$. C는 long long을 반환한다. Python 또는 C 중 하나로 함수를 작성하시오.

함수의 동작, 경계 처리, 시간 복잡도를 함께 작성하시오.

문제 99 정답과 해설

인접 선택 금지 최대 합

앞의 최적값 두 개로 마지막 원소의 선택·미선택을 비교한다.

해설

시간 O(n). 모두 음수여도 답 0을 유지한다.

문제 99 Python 구현

인접 선택 금지 최대 합

```python
def nonadjacent(a):
    prev2 = prev1 = 0
    for value in a:
        current = max(prev1, prev2 + value)
        prev2, prev1 = prev1, current
    return prev1
```

시간 O(n). 모두 음수여도 답 0을 유지한다.

문제 99 C 구현

인접 선택 금지 최대 합

```c
long long nonadjacent(const int a[], int n) {
    long long prev2=0, prev1=0;
    for (int i=0; i<n; ++i) {
        long long take=prev2+a[i];
        long long current=prev1>take ? prev1:take;
        prev2=prev1; prev1=current;
    }
    return prev1;
}
```

시간 O(n). 모두 음수여도 답 0을 유지한다.

문제 100

1≤행·열≤100, 칸 비용은 0..10^6이다. 왼쪽 위에서 오른쪽 아래까지 오른쪽·아래·오른쪽 아래 대각선으로 이동한다. 시작 칸은 전액, 오른쪽·아래로 도착하면 도착 칸 전액, 대각선으로 도착하면 도착 칸 비용//2를 낸다. 예: [[4,8],[6,5]]에서 대각선 경로 비용은 4+5//2=6이다. 최소 비용 grid_cost를 Python 또는 C로 작성하시오. C 열 폭은 100이다.

함수의 동작, 경계 처리, 시간 복잡도를 함께 작성하시오.

문제 100 정답과 해설

대각선 할인 격자

위·왼쪽은 전액, 왼쪽 위는 비용//2를 더해 최솟값을 선택한다.

해설

시작 칸은 4, 예시의 대각선 도착 칸은 5//2=2라서 총 6이다. 격자 밖 후보는 제외한다. 시간은 O(행×열)이다.

문제 100 Python 구현

대각선 할인 격자

```python
def grid_cost(a):
    rows, cols = len(a), len(a[0])
    dp = [[10**18]*cols for _ in range(rows)]
    dp[0][0] = a[0][0]
    for i in range(rows):
        for j in range(cols):
            if i: dp[i][j] = min(dp[i][j], dp[i-1][j]+a[i][j])
            if j: dp[i][j] = min(dp[i][j], dp[i][j-1]+a[i][j])
            if i and j:
                dp[i][j] = min(dp[i][j], dp[i-1][j-1]+a[i][j]//2)
    return dp[-1][-1]
```

시간 O(행×열). 시작 칸에는 할인하지 않는다.

문제 100 C 구현

대각선 할인 격자

```
long long grid_cost(const int a[][100], int rows, int cols,
                    long long dp[][100]) {
    for (int i=0; i<rows; ++i) {
        for (int j=0; j<cols; ++j) {
            long long best=1000000000000000000LL, t;
            if (i==0 && j==0) best=a[0][0];
            if (i) {t=dp[i-1][j]+a[i][j]; if(t<best) best=t;}
            if (j) {t=dp[i][j-1]+a[i][j]; if(t<best) best=t;}
            if (i && j) {
                t=dp[i-1][j-1]+a[i][j]/2; if(t<best) best=t;
            }
            dp[i][j]=best;
        }
    }
    return dp[rows-1][cols-1];
}
```

시간 O(행×열). 시작 칸에는 할인하지 않는다.