CSE 101
Discussion Section
Week 7

February 19 - February 26

Problem 1

Statement. Find the number of strings of length n that consist of digits/characters $0, 1, 2$ and 3 and that do not contain strings 01, 10 and 23 as substrings.

Problem 1

Statement. Find the number of strings of length n that consist of digits/characters $0, 1, 2$ and 3 and that do not contain strings 01, 10 and 23 as substrings.

Examples.

- $n = 1$: $0, 1, 2, 3$. The answer is 4.

Problem 1

Statement. Find the number of strings of length n that consist of digits/characters $0, 1, 2$ and 3 and that do not contain strings 01, 10 and 23 as substrings.

Examples.

- $n = 1$: $0, 1, 2, 3$. The answer is 4.
- $n = 2$: $00, 02, 03, 11, 12, 13, 20, 21, 22, 30, 31, 32, 33$. The answer is 13.

Problem 1

Statement. Find the number of strings of length n that consist of digits/characters $0, 1, 2$ and 3 and that do not contain strings 01, 10 and 23 as substrings.

Examples.

- $n = 1$: $0, 1, 2, 3$. The answer is 4.
- $n = 2$: $00, 02, 03, 11, 12, 13, 20, 21, 22, 30, 31, 32, 33$. The answer is 13.
- 021203 is a valid string.

Problem 1

Statement. Find the number of strings of length n that consist of digits/characters $0, 1, 2$ and 3 and that do not contain strings 01, 10 and 23 as substrings.

Examples.

- $n = 1$: $0, 1, 2, 3$. The answer is 4.
- $n = 2$: $00, 02, 03, 11, 12, 13, 20, 21, 22, 30, 31, 32, 33$. The answer is 13.
- 021203 is a valid string.
- 021**23**3 is not a valid string.

Problem 1

Solution.

- Let $T(n)$ be the number of valid strings of length n.

Problem 1

Solution.

- Let $T(n)$ be the number of valid strings of length n.
- In order to compute $T(n)$ we will use auxiliary function $f(i, n)$ – the number of valid strings of length n that end with digit i, $i \in \{0, 1, 2, 3\}$

Problem 1

Solution.

- Let $T(n)$ be the number of valid strings of length n.
- In order to compute $T(n)$ we will use auxiliary function $f(i, n)$ – the number of valid strings of length n that end with digit i, $i \in \{0, 1, 2, 3\}$
- We can see that $T(n) = f(0, n) + f(1, n) + f(2, n) + f(3, n)$.

Problem 1

Now, let's find values for $f(i, n)$ for $n > 0$:

- $f(0, n) = f(0, n-1) + f(2, n-1) + f(3, n-1)$. We take all valid strings of length $n-1$ that do not end with 1 and add 0 to them at the end.

Problem 1

Now, let's find values for $f(i, n)$ for $n > 0$:

- $f(0, n) = f(0, n-1) + f(2, n-1) + f(3, n-1)$. We take all valid strings of length $n-1$ that do not end with 1 and add 0 to them at the end.
- $f(1, n) = f(1, n-1) + f(2, n-1) + f(3, n-1)$. We take all valid strings of length $n-1$ that do not end with 0 and add 1 to them at the end.

Problem 1

Now, let's find values for $f(i, n)$ for $n > 0$:

- $f(0, n) = f(0, n-1) + f(2, n-1) + f(3, n-1)$. We take all valid strings of length $n-1$ that do not end with 1 and add 0 to them at the end.
- $f(1, n) = f(1, n-1) + f(2, n-1) + f(3, n-1)$. We take all valid strings of length $n-1$ that do not end with 0 and add 1 to them at the end.
- $f(2, n) = f(0, n-1) + f(1, n-1) + f(2, n-1) + f(3, n-1)$. We take all valid strings of length $n-1$ and add 2 to them at the end.

Problem 1

Now, let's find values for $f(i, n)$ for $n > 0$:

- $f(0, n) = f(0, n-1) + f(2, n-1) + f(3, n-1)$. We take all valid strings of length $n-1$ that do not end with 1 and add 0 to them at the end.
- $f(1, n) = f(1, n-1) + f(2, n-1) + f(3, n-1)$. We take all valid strings of length $n-1$ that do not end with 0 and add 1 to them at the end.
- $f(2, n) = f(0, n-1) + f(1, n-1) + f(2, n-1) + f(3, n-1)$. We take all valid strings of length $n-1$ and add 2 to them at the end.
- $f(3, n) = f(0, n-1) + f(1, n-1) + f(3, n-1)$. We take all valid strings of length $n-1$ that do not end with 2 and add 3 to them at the end.

Problem 1

So we have:

$$
\begin{aligned}
f(0,n) &= f(0,n-1) + f(2,n-1) + f(3,n-1) \\
f(1,n) &= f(1,n-1) + f(2,n-1) + f(3,n-1) \\
f(2,n) &= f(0,n-1) + f(1,n-1) + f(2,n-1) + f(3,n-1) \\
f(3,n) &= f(0,n-1) + f(1,n-1) + f(3,n-1)
\end{aligned}
$$

Base case:

$$f(0,0) = f(1,0) = f(2,0) = f(3,0) = 1$$

The answer is:

$$T(n) = f(0,n) + f(1,n) + f(2,n) + f(3,n)$$

Problem 1

Both the time complexity and the space complexity of the solution is $\mathcal{O}(n)$.

Can we optimize it further?

Problem 1

Both the time complexity and the space complexity of the solution is $\mathcal{O}(n)$.

Can we optimize it further?

The value $f(i, n)$ only depends on values $f(x, n-1)$. If we consider $f(i, n)$ as a two dimensional array, then the values at column n depend only on values at column $n-1$. Thus, there is no need to store all the values in the array to find $f(i, n)$. We only need values for the previous column.

Problem 1

Base case:

$$f(0,0) = f(1,0) = f(2,0) = f(3,0) = 0$$

Updated formulas ($n > 0$):

$$\begin{aligned}
f(0, n \bmod 2) &= f(0,(n-1) \bmod 2) + f(2,(n-1) \bmod 2) \\
&+ f(3,(n-1) \bmod 2) \\
f(1, n \bmod 2) &= f(1,(n-1) \bmod 2) + f(2,(n-1) \bmod 2) \\
&+ f(3,(n-1) \bmod 2) \\
f(2, n \bmod 2) &= f(0,(n-1) \bmod 2) + f(1,(n-1) \bmod 2) \\
&+ f(2,(n-1) \bmod 2) + f(3,(n-1) \bmod 2) \\
f(3, n \bmod 2) &= f(0,(n-1) \bmod 2) + f(1,(n-1) \bmod 2) \\
&+ f(3,(n-1) \bmod 2)
\end{aligned}$$

The answer is
$f(0, n \bmod 2) + f(1, n \bmod 2) + f(2, n \bmod 2) + f(3, n \bmod 2)$.

Problem 1

Now our solution has $\mathcal{O}(n)$ time complexity and $\mathcal{O}(1)$ space complexity.

The knapsack problem where we can take each item once has the following formula:

$$f(i,k) = max(f(i-1,k), f(i-1,k-w_i) + p_i)$$

In a similar way we can optimize the space complexity by only storing values $f(i-1,k)$ (for all k) to compute values $f(i,k)$.

Problem 2

You are given a sequence of n integer numbers $A = (a_1, a_2, ..., a_n)$. Find a sub-sequence A' of the sequence A, such that the sum of all elements in A' is maximized and A' can't contain any neighboring elements at the same time.

Problem 2

Solution. Let's define two functions $f(n)$ and $g(n)$:

- $f(n)$ – is the sum of the optimal sub-sequence A' in sequence A if element a_n is included to A'.

Problem 2

Solution. Let's define two functions $f(n)$ and $g(n)$:

- $f(n)$ – is the sum of the optimal sub-sequence A' in sequence A if element a_n is included to A'.
- $g(n)$ – is the sum of the optimal sub-sequence A' in sequence A if element a_n is not included to A'.

Problem 2

Solution. Let's define two functions $f(n)$ and $g(n)$:

- $f(n)$ – is the sum of the optimal sub-sequence A' in sequence A if element a_n is included to A'.
- $g(n)$ – is the sum of the optimal sub-sequence A' in sequence A if element a_n is not included to A'.
- Base case: $f(1) = a_1$, $g(1) = 0$

Problem 2

For $n > 1$:

- $f(n) = g(n-1) + a_n$. We know that we must include a_n to the sub-sequence A', so we add its value. Then, if a_n is in the sub-sequence, we can't have a_{n-1} there. So we find the optimal sub-sequence of a sequence that consists of first $n-1$ elements and doesn't include element a_{n-1}.

Problem 2

For $n > 1$:

- $f(n) = g(n-1) + a_n$. We know that we must include a_n to the sub-sequence A', so we add its value. Then, if a_n is in the sub-sequence, we can't have a_{n-1} there. So we find the optimal sub-sequence of a sequence that consists of first $n-1$ elements and doesn't include element a_{n-1}.
- $g(n) = max(f(n-1), g(n-1))$. We can't add a_n to the optimal sub-sequence. There are no more restrictions except this. Thus, we find the optimal sub-sequence among the first $n-1$ elements.

Problem 2

For $n > 1$:

- $f(n) = g(n-1) + a_n$. We know that we must include a_n to the sub-sequence A', so we add its value. Then, if a_n is in the sub-sequence, we can't have a_{n-1} there. So we find the optimal sub-sequence of a sequence that consists of first $n-1$ elements and doesn't include element a_{n-1}.
- $g(n) = max(f(n-1), g(n-1))$. We can't add a_n to the optimal sub-sequence. There are no more restrictions except this. Thus, we find the optimal sub-sequence among the first $n-1$ elements.

The answer is $max(f(n), g(n))$.

Problem 2

So we have ($n > 1$):

$$f(n) = g(n-1) + a_n$$
$$g(n) = max(f(n-1), g(n-1))$$

Base case:

$$f(1) = a_1$$
$$g(1) = 0$$

Problem 2

So we have ($n > 1$):

$$f(n) = g(n-1) + a_n$$
$$g(n) = max(f(n-1), g(n-1))$$

Base case:

$$f(1) = a_1$$
$$g(1) = 0$$

Right now, both the time complexity and space complexity of the solution is $\mathcal{O}(n)$. We can optimize it in a similar way to Problem 1.

Problem 2

After the optimization we get ($n > 1$):

$$f(n \bmod 2) = g((n-1) \bmod 2) + a_n$$
$$g(n \bmod 2) = max(f((n-1) \bmod 2), g((n-1) \bmod 2))$$

Base case:

$$f(1) = a_1$$
$$g(1) = 0$$

Problem 2

After the optimization we get ($n > 1$):

$$f(n \bmod 2) = g((n-1) \bmod 2) + a_n$$
$$g(n \bmod 2) = max(f((n-1) \bmod 2), g((n-1) \bmod 2))$$

Base case:

$$f(1) = a_1$$
$$g(1) = 0$$

The answer is $max(f(n \bmod 2), g(n \bmod 2))$.

Problem 2

After the optimization we get ($n > 1$):

$$f(n \bmod 2) = g((n-1) \bmod 2) + a_n$$
$$g(n \bmod 2) = max(f((n-1) \bmod 2), g((n-1) \bmod 2))$$

Base case:

$$f(1) = a_1$$
$$g(1) = 0$$

The answer is $max(f(n \bmod 2), g(n \bmod 2))$.

Now, the time complexity is $\mathcal{O}(n)$ and the space complexity is $\mathcal{O}(1)$.