Coin Row Problems

In any coin row problem, we are given a row of n coins, each of which has some value. Our goal is to select a maximum weight *legal* subset of the coins, where the definition of legality is different for each version. Our examples can be solved using dynamic programming.

For each problem, let the coins be numbered $1 \ldots n$, and let $V[i]$ be the value of the i^{th} coin. For simplicity, we assume that $V[i] > 0$ for each i.

The Original Version

Define a subset to be legal if it contains no two consecutive coins of the row. We give two dynamic programs.

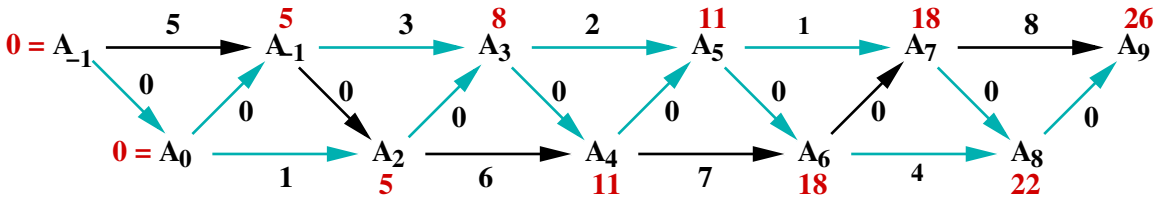
Best subsequence up to some point

Let $A[i]$ be the maximum weight of any legal subsequence of the first i coins. For convenience we introduce dummy values to make our code more uniform; We compute $A[i]$ for $-1 \leq i \leq n$. as follows.

Program 1:

```
A[-1] = 0; // dummy value
A[0] = 0; // dummy value
for(int i = 1; i <= n; i++)
 A[i] = max(A[i-1],A[i-2]+V[i]);
return A[n];
```

As usual in dynamic programming, the coint row problem reduces to a shortest path problem in a weighted directed graph. We use an explicit example sequence of the $\{V[i]\}$: 5, 1, 3, 6, 2, 7, 1, 4, 8. The maximum total legal subsequence is 5, 6, 7, 8.



The Graph Corresponding to Program 1

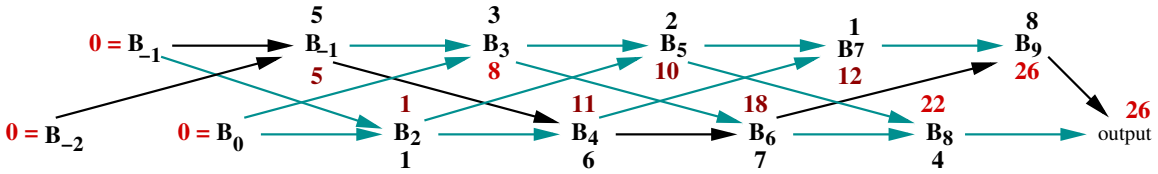
Best subsequence ending at some point

Let $B[i]$ be the maximum weight of any legal subsequence which ends at the i^{th} coin.

Program 2:

```
B[-2] = 0; // dummy value
B[-1] = 0; // dummy value
```

```
B[0] = 0; // dummy value
for(int i = 1; i <= n; i++)
 B[i] = V[i] + max(B[i-2],B[i-3]);
return max(B[n],B[n-1]);
```



The Graph Corresponding to Program 2

The Version on Test 2

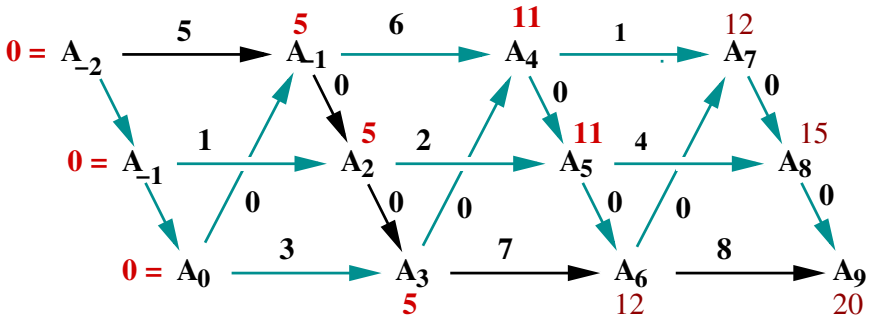
A subset is legal if any two coins in the set must have at least two coins between them in the original row. Again, we use the sequence: 5, 1, 3, 6, 2, 7, 1, 4, 8. As before, we have two dynamic programs.

Best subsequence up to some point

Let $A[i]$ be the maximum weight of any legal subsequence of the first i coins.

Program 3

```
A[-2] = 0; // dummy value
A[-1] = 0; // dummy value
A[0] = 0; // dummy value
for(int i = 1; i <= n; i++)
 A[i] = max(A[i-1],A[i-2],A[i-3]+V[i]);
return A[n];
```



The Graph Corresponding to Program 3

Best subsequence ending at some point

Let $B[i]$ be the maximum weight of any legal subsequence which ends at the i^{th} coin.

Program 4

```
B[-4] = 0; // dummy value
B[-3] = 0; // dummy value
B[-2] = 0; // dummy value
B[-1] = 0; // dummy value
B[0] = 0; // dummy value
for(int i = 1; i <= n; i++)
 B[i] = V[i] + max(B[i-3],B[i-4],B[i-5]);
return max(B[n],B[n-1],B[n-2]);
```

The maximum total legal subsequence is 5, 7, 8.

Do you understand these programs?

The graph for Program 4 is more complex than the others. Can you draw it?

A Harder Version

The coin row problem gets harder as we pick more and more complex definitions of legality. For this version, we define a subset to be legal if it does not have any three consecutive coins of the original row.

Writing the dynamic program for this version of the coin row problem is considerably harder than for the previous two.