N-COIN PROBLEM

PRASANG UPADHYAYA

1. Solution

The problem solved is a general n coins problem. One of the coins is a counterfeit coin. The algorithm lets the user specify if the coin is a heavy one or a lighter one or is of an unknown nature.

1.1. Basic algorithm.

A dynamic programming based approach has been used to compute the optimal strategies. The optimal solution at any stage makes use of the optimal solutions to its descendent subproblems.

A state (which is the input to the DP algorithm) is defined as a 4-tuple of (l, h, u, k). Here,

- **l:** Suspected lighter coins.
- **h:** Suspected heavier coins.
- **u:** Coins which contain the counterfeit coins.
- **k:** Coins which are known to be normal.

Note that, either $u = 0$ or $l + h > 0$. This is because there is only one counterfeit coin. If we have non-zero values for l and/or h, the remaining coins have to be normal. Thus, the actual set of legal states is smaller than n^4, where n is the number of coins given. Also, the sum of all the coin set has to be less than or equal to n. This further reduces the number of legal states.

Given a state S, we make two piles of equal sizes: $L = (l_1, h_1, u_1, k_1)$ and $R = (l_2, h_2, u_2, k_2)$. All variables are non-negative integers. These piles satisfy the following constraints:

$$\begin{aligned} l_2 + h_2 + u_2 + k_2 &= l_1 + h_1 + u_1 + k_1 \\ l_1 + l_2 &\leq l \\ h_1 + h_2 &\leq h \\ u_1 + u_2 &\leq u \\ k_1 + k_2 &\leq k \\ min\{k_1, k_2\} &= 0 \end{aligned}$$

The first condition above ensures that each weighing has an equal number of coins on both sides. Unequal weighing is not likely to give any information and hence is won't reduce the belief space.

All possible piles are enumerated and the optimal plan sizes for those enumerations are computed. The size of the plan for present state is 1+max(size of optimal plans for children).

Significant overlap among the descendants of different plans lets the algorithm avoid repeated computations.

The subproblem for a given pile selection is defined in as follows:

1.1.1. $u = 0$. This case covers those situations where we know that the counterfeit coin is in either the lighter coin set (l) or in the heavier one (h). Note that $u_1 = 0$ and $u_2 = 0$ Given a pile division, there are three possible outcomes:

- **Left $<$ Right:** Counterfeit coin is light and in l_1 or is heavy and in h_2. All other coins are normal.
- **Equal:** Counterfeit coin is light and in $l - l_1 - l_2$ or is heavy and in $h - h_1 - h_2$. All other coins are normal.
- **Left $>$ Right:** Counterfeit coin is light and in l_2 or heavy and in h_1. All other coins are normal.

The optimal solutions for all the three subproblems is computed recursively using *memoization*.

1.1.2. $u > 0$. This case covers those situations where we haven't yet figured out the lighter coin set (l) and the heavier coin set (h). Note that $l_1 = l_2 = h_1 = h_2 = 0$. Given a pile division, there are three possible outcomes:

- **Left $<$ Right:** Counterfeit coin is light and in u_1 or is heavy and in u_2. All other coins are normal.
- **Equal:** u_1 and u_2 are also normal coins. Counterfeit coin is in $u - u_1 - u_2$ and could be either heavy or light.
- **Left $>$ Right:** Counterfeit coin is light and in u_2 or heavy and in u_1. All other coins are normal.

The optimal solutions for all the three subproblems is computed recursively using *memoization*.

1.2. **Optimizations.** The following two techniques were implemented to improve running time.

- For a given state $S = (l, h, u, k)$ the optimal plan might use atmost k_o coins from k. Thus, the obtained optimal plan for state S would also be the same for all states $S_i = (l, h, u, i)$ where $k_o \leq i \leq k$.
- Symmetrical states with respect to l and h have symmetrical solutions. Thus, if for a state $D = (l, h, u, k)$, the optimal piles are given as $L_o = (l_1, h_1, u_1, k_1)$ and $R_o = (l_2, h_2, u_2, k_2)$, the the optimal pile for state $S_{sym} = (h, l, u, k)$ is given by $L_{sym} = (h_1, l_1, u_1, k_1)$ and $R_{sym} = (h_2, l_2, u_2, k_2)$

1.3. **Heuristic.** Given a state $S = (l, h, u, k)$, a lower bound on the size of the plan can be specified as $\lceil \log_3 (l + h + 2u) \rceil$. The factor of 2 arises because the coins in set represented by u can be either lighter or heavier than normal.

This heuristic can be used to prune out states which won't improve the current best solution.

1.4. **Observations.** The performance of the algorithms was evaluated by computing the total number of pile-splits that were enumerated and explored. Using the optimizations and the pruning heuristic, the actual number of splits that were explored got reduced.

The table 1 lists the total number of splits that different algorithms explored. Table 2 lists the number of sub-problems for which optimal plans were known, in the optimal solution to the original problem. Usually, the values in the second table are smaller than the values in the first table. This is because many different enumerations lead to the same subproblems eventually, and also because many different enumerations don't lead to optimal solution, but are checked nevertheless.

In the tables, the convention followed is as follows. *coins* represent the number of coins. There is a single counterfeit coin which can be heavier or lighter than normal. The binary string $b_1b_2b_3$ represents the algorithm: $b_1 = 0$ implies that the pruning heuristic wasn't used, and $b_1 = 1$ implies that the pruning heuristic is used; b_2 represents the symmetrical state optimization and b_3 represents the other optimization (referred to as min-k from now on).

The min-k optimization doesn't lead to any improvement as far as the number of splits evaluated is concerned. A possible explanation for this can be that the actual implementation of the algorithm implicitly chooses the minimum k for the subproblems. This is because, while invoking the function of a subproblem the value of k is taken to be $min\{l + h, u, N - l - h - u\}$, where N is the total number of coins in the problem.

The symmetric optimization reduces the number of splits explored by any algorithm by a factor of about a third. This is because, in many cases, the symmetric states optimal solutions are required to be computed later, but the optimization pre-computes the value without actually enumerating all splits.

The pruning heuristic substantially reduces the number of splits considered. The reduction seems to depend on the size of the optimal plan. A distinct jump in the values occurs near those coin sizes where the plan size increases by one.

For the case where we know if the coin is heavier or lighter, the symmetry optimizations don't lead to any improvement because the symmetrical states are not-reachable if the coin's bias is known beforehand. Thus, the normal algorithm automatically avoids the symmetrical states.

1.5. **Executing the program.** Compile the c++ program **coins.cpp**. Run and follow the instructions.

Table 1. Enumerations of states for different algorithms.

coins	000	001	010	011	100	101	110	111	Plan-size
3	7	7	7	7	3	3	3	3	2
4	30	30	25	25	7	7	7	7	3
5	79	79	59	59	12	12	11	11	3
6	183	183	129	129	26	26	23	23	3
7	323	323	222	222	44	44	36	36	3
8	643	643	419	419	97	97	76	76	3
9	960	960	623	623	133	133	100	100	3
10	1751	1751	1086	1086	258	258	157	157	3
11	2385	2385	1484	1484	272	272	192	192	3
12	4078	4078	2444	2444	390	390	252	252	3
13	5235	5235	3156	3156	1441	1441	1328	1328	4
14	8501	8501	4968	4968	2783	2783	2630	2630	4
15	10481	10481	6167	6167	3648	3648	3455	3455	4
16	16307	16307	9350	9350	8074	8074	5482	5482	4
17	19520	19520	11268	11268	10842	10842	7098	7098	4
18	29293	29293	16547	16547	16952	16952	10128	10128	4
19	34295	34295	19497	19497	20313	20313	12071	12071	4
20	49898	49898	27851	27851	28613	28613	16086	16086	4

Table 2. Number of states explored in the optimal plan for different algorithms.

coins	000	001	010	011	100	101	110	111	Plan-size
3	5	7	5	7	5	7	5	7	2
4	9	18	9	18	8	13	9	15	3
5	12	28	12	28	11	17	12	18	3
6	18	49	18	49	16	33	18	41	3
7	21	63	21	63	21	46	21	48	3
8	29	99	29	99	28	70	29	76	3
9	32	118	32	118	32	82	32	84	3
10	42	171	42	171	41	118	41	120	3
11	45	196	45	196	42	128	42	136	3
12	57	269	57	269	49	167	49	179	3
13	60	301	60	301	56	198	60	225	4
14	74	397	74	397	66	260	74	307	4
15	77	436	77	436	69	289	77	342	4
16	93	558	93	558	85	397	91	460	4
17	96	607	96	607	90	450	96	517	4
18	114	758	114	758	106	579	112	662	4
19	117	817	117	817	111	648	117	733	4
20	137	1000	137	1000	130	817	135	902	4