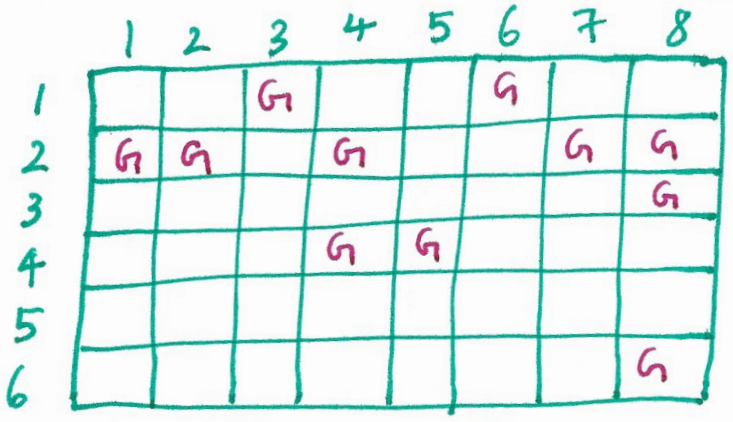
Design & Analysis of Algorithms

Given a problem, design an algorithm to solve it.

Prove its correctness/optimality.

Analyze its complexity

Robotic Coin Collection



Robots start at top left and move to bottom right only making right/down moves

Find a path that results in the maximum # coins picked up.

We can use dynamic programming

Dynamic Programming Solution

Subproblems: $F(i,j)$ largest # coins robot can bring to and including cell (i,j)

$C_{ij} = 1$ if there is a coin on cell (i,j) and 0 otherwise

Recurrence: $F(i,j) = \max\left(F(i-1,j), F(i,j-1)\right) + C_{ij}$

for $1 \le i \le n$, $1 \le j \le m$

$F(0,j) = 0$ for $1 \le j \le m$

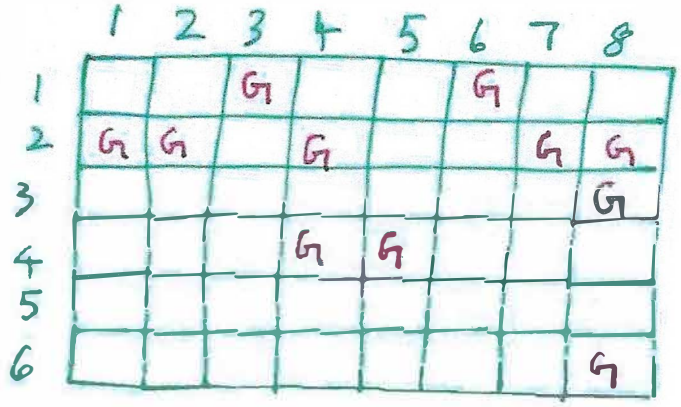
$F(i,0) = 0$ for $1 \le i \le n$

Solution: $F(n,m)$

Time complexity: $\theta(nm)$

Space complexity: $\theta(nm)$

Robotic Coin Collection II



Robots start at top left and move to bottom right only making right/down moves.

Minimize # robots required to pick up ALL coins.

Dynamic Programming Strategy Greedily Applied

Send 1st robot and collect maximum # coins. Coins are removed.
Second robot collects max # coins ...
Does this minimize the # robots? NO

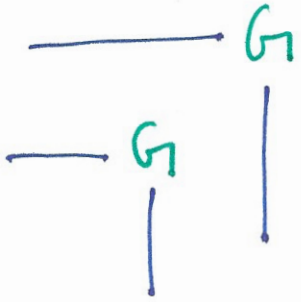
In example above, first robot collects 7 coins: 5 in row 2 and coins in column 8
Need 2 more robots for remaining coins.

Can do better!

(1, 2) (2, 2) (4, 2) (4, 4) (5, 4) (8, 6) 6 coins

(3, 1) (6, 1) (7, 2) (8, 2) (8, 3) 5 coins

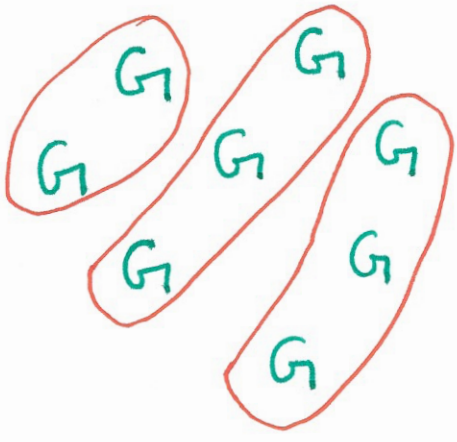
Observation



Two occupied cells are disjoint if one is to the bottom left of another. Cells that are disjoint from each other need different robots.

The largest maximal set of disjoint cells
⇒ a lower bound on the # robots reqd.
Equal to its size.

Maximal Sets



maximal sets, ie. cannot grow them by adding another cell that is disjoint to ALL cells in existing set.

If we can design an algorithm that shrinks all the largest maximal sets by 1 with each robot's path, algorithm is optimal!

INTUITION: PEEL OFF BOTTOMMOST LAYER ITERATIVELY

FORMALIZATION

Set of robot paths $\{P_i\}$ should satisfy the following two properties:

1) Paths cover all the coins
2) For every path P_i there exists a coin $c_j \in P_i$ such that the $\{c_j\}$ form a disjoint / independent set

Then, we can claim that $\{P_i\}$ is a smallest path set.

NOTATION

C : set of uncollected coins

$C_{x \in X}$: Coins in C such that x coordinate $\in X$

Similarly, $C_{y \in Y}$, $C_{x \in X, y \in Y}$

$C_{y \leq y_1}$, $C_{x \leq x_1, y \leq y_1}$

PEEL ALGORITHM PSEUDOCODE (COURTESY: JUN WAN)

Robot counter $i \leftarrow 0$, Path set $P = \emptyset$

while $C \neq \emptyset$:

Create a new robot

$x_1 = 0$, $y_1 = 1$

while $C_{x>x_1, y \geq y_1} \neq \emptyset$:

Find the coin with minimum $x \in C_{x>x_1, y \geq y_1}$ and set x_1 to x-coordinate value of coin.

Robot collects coins $T = C_{x=x_1, y \geq y_1}$

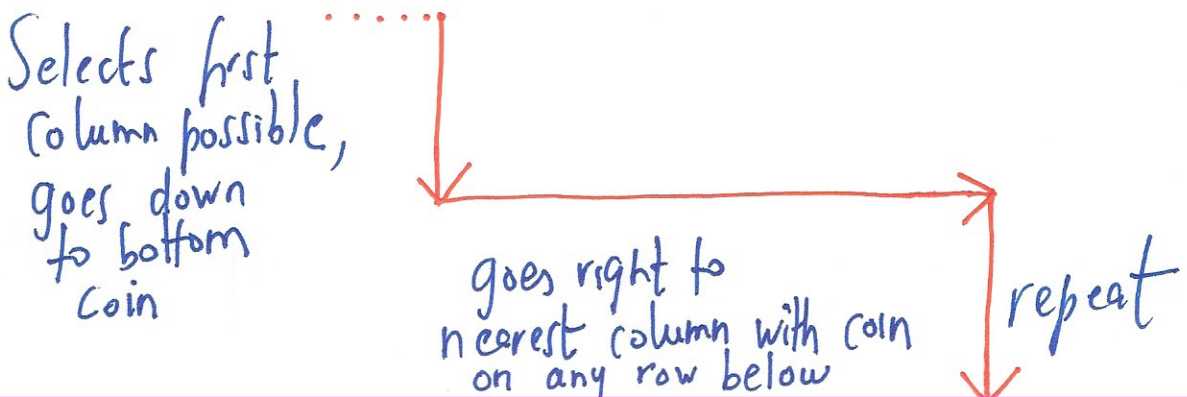
Update $y_1 = \max_y C_{x=x_1, y \geq y_1}$, $C = C - T$

$i = i + 1$

Denote P_i as the path created by this robot

$P = P \cup P_i$

Return P



Proof

Property 1 satisfied since algorithm only terminates when all coins are collected.

For the second property, we will show that for any two paths P_i, P_j $i<j$ and each coin $c \in P_j$ there exists a coin $c' \in P_i$ that is disjoint from c.

Observe that whenever we create a new path we include the entire column with minimum x-coordinate. Therefore, there must exist coins in P_i whose x-coordinate is less than x-coordinate of c, call it c_x. Denote c' as the lowest coin on the column nearest to c coin's left.

CLAIM: c' is below c, i.e., $c'_x < c_x, c'_y > c_y$.

true by definition

If $c'_y \leq c_y$, then algorithm will include c' in P_i. This contradicts our assumption that $c \in P_j$.

Proof (contd.)

Now we show how to construct a disjoint set as in Property 2.

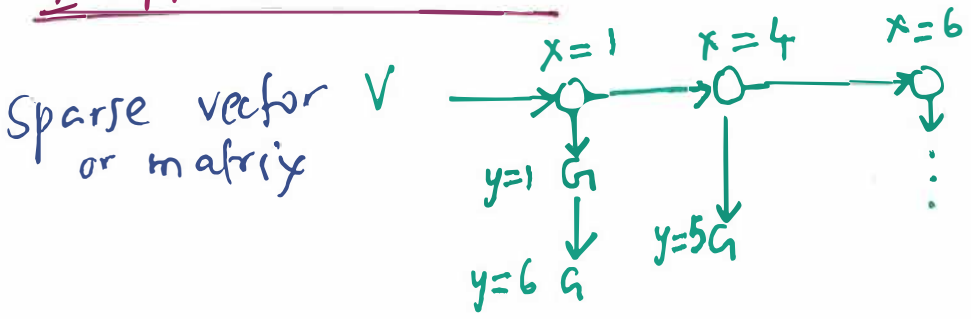
- Find a coin $C_r \in P_r$ ← number of robots required
- Find a coin $C_{r-1} \in P_{r-1}$ disjoint from C_r
- Find a coin $C_{r-2} \in P_{r-2}$ disjoint from C_{r-1}
  C_{r-2} is disjoint from C_r due to transitivity of $<$.
- Repeat until $C_1 \in P_1$ disjoint from C_2 and all other $C_{j>2}$

$\{C_1, .. C_r\}$ forms a disjoint set. ☒

Runtime Analysis

Depends on data structures used.
Outer loop iterations $= r$
With matrix representation, at least $\theta(nm)$ for each robot.
Can we do better?

IMPROVEMENTS



$\Theta(k)$ space, where k is the #coins.

Original algorithm: $\Theta(rk)$ time

(r: number of robots)

Can we do better?

Order coins: columns from left to right, each column top to bottom ↓↓↓↓...

$y_0 = \infty$ $r = 0$

For the first coin $c=(x,y)$, set $y_1 = y$, $r = 1$

For remaining coins $c = (x,y)$:

if $y \in [y_i, y_{i-1})$ for some $1 \le i \le r$, then assign c to path i, set $y_i = y$

if y is smaller than all y_i, $1 \le i \le r$, create a new robot/path. $r = r+1$. $y_r = y$.

Algorithm assigns coins in one pass.

Runs in $\Theta(k \log r)$ time with appropriate data structure for intervals.