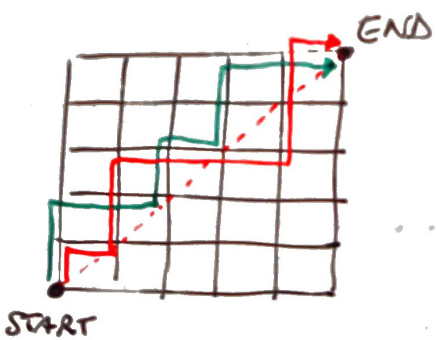
P(T)VZZLE 14

YOU ARE ON A $N \times N$ GRID OF INTEGERS IN THE POSITION $(0,0)$ AND YOU WANT TO GO TO THE POSITION (N,N). ONLY STEPS TOWARD NORTH AND EAST ARE ALLOWED. HOW MANY DIFFERENT PATHS YOU CAN BUILD THAT LIVE TOTALLY ABOVE (OR BELOW) THE DIAGONAL CONNECTING $(0,0)$ TO (N,N)? CAN YOU DESIGN AN EFFICIENT (IN TIME AND SPACE) ALGORITHM TO FULLY CREATE ALL THOSE PATHS?

SOLUTION

FIRST OF ALL LET'S MAKE CLEAR WHAT ARE THE PATHS WE ARE LOOKING FOR. HERE A SIMPLE EXAMPLE



SO THE FIRST QUESTION IS TO FIND HOW MANY GREEN PATHS WE CAN BUILD.

AS FIRST OBSERVATION ALL PATHS WILL BE COMPOSED BY $2N$ STEPS.

BECAUSE WE CAN MOVE ONLY NORTH (N) AND EAST (E), THE $2N$ STEPS WILL BE SPLIT AS

$$\begin{cases} N \text{ STEPS} \rightarrow \text{NORTH} \\ N \text{ STEPS} \rightarrow \text{EAST} \end{cases}$$

ALL POSSIBLE PATHS, GOOD AND BAD, CONNECTING ORIGIN $(0,0)$ AND DESTINATION (N,N) ARE EASILY COUNTED AS

$$N_{TOT} = \binom{2N}{N}$$

INDEED YOU CAN THINK THAT AS AN ALLOCATION PROBLEM OF N SYMBOLS ($\uparrow$ = NORTH) INTO A $2N$ POSSIBLE SLOTS.

IF WE INDICATE

$\hat{N}$ = # GOOD PATHS

$\tilde{N}$ = # BAD PATHS

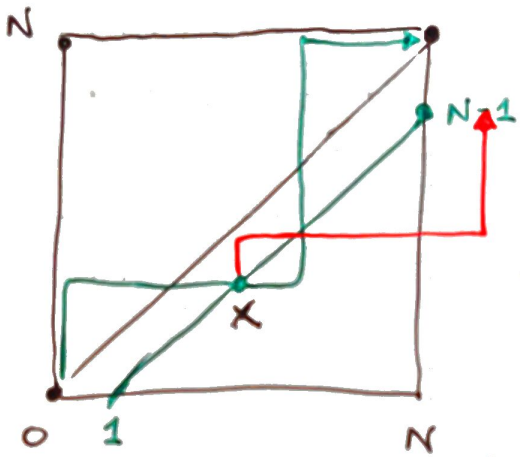
WE CAN WRITE

$$\hat{N} = N_{TOT} - \tilde{N}$$

SO OUR GOAL NOW IS TO FIND $\tilde{N}$.

FOR DOING THAT WE WILL SHOW A BIJECTION BETWEEN $\tilde{N}$ AND A MORE SIMPLE COUNTING PROBLEM.

LET'S START CONSIDERING THE SUB-DIAGONAL OF THE (N,N) GRID, I.E. THE DIAGONAL CONNECTING (1, 0) TO (N, N-1)



ALL BAD PATHS WILL INTERCEPT THIS SUB-DIAGONAL IN A POINT X (I MEAN THE FIRST INTERSECTION POINT). WE TAKE THE PORTION OF THE PATH AFTER X AND WE WILL REFLECT IT WITH RESPECT THE SUB-DIAGONAL.

IN THAT WAY THE PATH WILL REACH THE POINT $(N+1, N-1)$ AND THIS WILL HAPPEN FOR ALL BAD PATHS. SO CONSIDERING THE GRID $(N+1, N-1)$ ALL PATHS CONNECTING $(0,0)$ TO $(N+1, N-1)$ WILL TOUCH THE SUB-DIAGONAL, SO ALL OF THEM ARE IN BIJECTION WITH THE BAD PATHS OF THE SQUARED GRID.
THE PATH OF THE RECTANGULAR GRID ARE SIMPLY GIVEN BY

$$\tilde{N} = \binom{2N}{N-1} \equiv \binom{2N}{N+1}$$

AS A CONSEQUENCE THE REQUESTED GOOD PATHS ARE

$$\hat{N} = N_{TOT} - \hat{N} = \binom{2N}{N} - \binom{2N}{N-1}$$

$$= \frac{(2N)!}{N!\,N!} - \frac{(2N)!}{(N-1)!\,(N+1)!} =$$

$$= \frac{(2N)!}{N!\,N!} - \frac{(2N)!\,N}{(N+1)\,N!\,N!} =$$

$$= \frac{N+1-N}{N+1} \cdot \frac{(2N)!}{N!N!} = \frac{1}{N+1} \frac{(2N)!}{N!N!}$$

$$\Rightarrow \boxed{\hat{N} = \frac{1}{N+1}\binom{2N}{N}}$$

THIS NUMBER IS KNOWN IN COMBINATORICS AS CATALAN NUMBER C_N.

LET'S GO NOW ON THE SECOND QUESTION. FIND AN ALGORITHM TO CREATE ALL C_N PATHS, IN AN EFFICIENT WAY.

NOTE: BECAUSE WE WANT TO CREATE ALL C_N PATHS, CLEARLY DESIGNING A SEQUENTIAL ALGORITHM WILL NEED $O(C_N)$ TIME THAT EXPLODES IMMEDIATELY. SO THE GOAL IS TO BE ABLE TO CREATE A PARALLEL ALGORITHM.

A SIMPLE ~~RECURSIVE~~ ALGORITHM WORKING IN $O(C_N)$ TIME AND $O(C_N)$ SPACE, SO NOT THE ONE WE ARE LOOKING FOR, IS GIVEN BY:

NORTH EAST
1 0

PATH : { LIST() ← LIST OF 1 AND 0
NORTH-COUNTER ← # OF NORTH STEPS.

PATH ← { (1)
1
PATHLIST ← PATH

```
BUILDPATH ( PATHLIST)
    WHILE PATHLIST ISNOT EMPTY
        PATH ← PATHLIST . POP()
        NEW PATH P_N ← EXPAND NORTH (PATH)
        NEW PATH P_E ← EXPAND EAST (PATH)
        PATHLIST . ADD (P_N)
        PATHLIST . ADD (P_E)
```

THE IDEA IS THAT ALL PATHS WILL START WITH A NORTH STEP, SO

IN THE PATHLIST ARE STORED PARTIAL PATHS AND EVERY TIME ONE OF THEM IS AUGMENTED CREATING AT MOST 2 NEW PATHS TAKING IT AND EXPANDING NORTH IF POSSIBLE (MAX NORTH STEPS IS N, THAT'S THE REASON FOR HAVING THE COUNTER INTO THE PATH OBJECT) AND EXPANDING EAST IF POSSIBLE (EAST STEPS MUST BE LESS OF NORTH STEPS IN ORDER TO EXPAND)

```
EXPANDNORTH(PATH)
    IF PATH.NORTHCOUNTER < N
        PATH.NORTH_COUNTER += 1
        PATH.LIST.ADD(1)
    ELSE IF PATH.LIST.COUNT == 2N
        PRINT(PATH.LIST())
        PATH <-- ϕ
    RETURN PATH
```

ok

THIS MUST NEVER HAPPEN BECAUSE THE PATH WILL END WITH EAST!

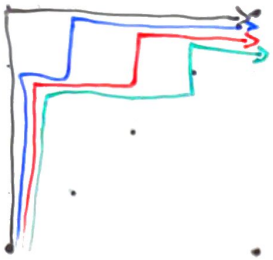
```
EXPAND EAST(PATH)
    IF  PATH.LIST.COUNT() > PATH.NORTH_COUNTER
        PATH.LIST.ADD(0)
    ELSE IF  PATH.LIST.COUNT() == 2N
        PRINT(PATH.LIST())
        PATH ← ϕ
    RETURN PATH
```

JUST TO HAVE AN EXAMPLE IN OUR MIND, LET'S SEE THE CASE $N=4$

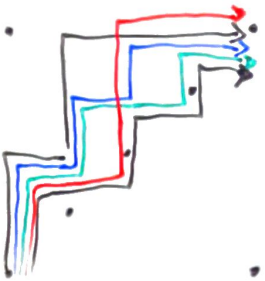
$$C_4 = \frac{1}{N+1}\binom{2N}{N} = \frac{1}{5}\binom{8}{4} = \frac{8!}{5 \cdot 4!\,4!}$$

$$= \frac{8 \cdot 7 \cdot 6 \cdot \cancel{5} \cdot \cancel{4!}}{\cancel{5} \cdot 4!\,\cancel{4!}} = 14$$

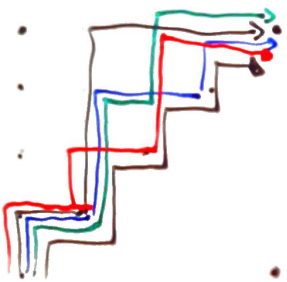
AND THE PATHS ARE:



NNNNEEEE
NNNENEEE
NNNEENEE
NNNEEENE



NNENNEEE
NNENENEE
NNENEENE
NNEENNEE
NNEENENE



NENNNEEE
NENNEENE
NENNENEE
NENENENE
NENENNEE

THE PREVIOUS SIMPLE ALGORITHM WILL WORK AS FOLLOWING:

- PATHLIST ← ("N")
- PATHLIST ← ("NN", "NE")
- PATHLIST ← ("NE", "NNN", "NNE")
- PATHLIST ← ("NNN", "NNE", "NEN")
- PATHLIST ← ("NNE", "NEN", "NNNN", "NNNE")
- PATHLIST ← ("NEN", "NNNN", "NNNE", "NNEN" "NNEE")
- PATHLIST ← ("NNNN", "NNNE", "NNEN", "NNEE", "NENN", "NENE")
- PATHLIST ← ("NNNE, NNEN, NNEE, NENN, NENE, NNNNE)
- (NNEN, NNEE, NENN, NENE, NNNNE NNNEN, NNNEE)
- (NNEE, NENN, NENE, NNNNE, NNNEN NNNEE, NNENN, NNENE)
- (NENN, NENE, NNNNE, NNNEN, NNNEE, NNENN, NNENE, NNEEN)
- (NENE, NNNNE, NNNEN, NNNEE, NNENN, NNENE, NNEEN, NENNN, NENNE)
- (NNNNE, NNNEN, NNNEE, NNENN, NNENE, NNEEN, NENNN, NENNE, NENEN)

- (NNNEN, NNNEE, NNENN, NNENE, NNEEN, NENNN, NENNE, NENEN, NNNNEE)
- (NNNEE, NNENN, NNENE, NNEEN, NENNN, NENNE, NENEN, NNNNEE, NNNEEN, NNNEEE)
- (NNENN, NNENE, NNEEN, NENNN, NENNE, NENEN, NNNNEE, NNNENE, NNNEEE, NNNEEN)
- (NNENE, NNEEN, NENNN, NENNE, NENEN NNNNEE, NNNENE, NNNEEE, NNNEEN, NNENNE)
- (NNEEN, NENNN, NENNE, NENEN, NNNNEE, NNNENE, NNNEEE, NNNEEN, NNENNE, NNENEN, NNENEE)
- (NENNN, NENNE, NENEN, NNNNEE, NNNENE, NNNEEE, NNNEEN, NNENNE, NNENEN, NNENEE, NNEENN, NNEENE)
- (NENNE, NENEN, NNNNEE, NNNENE, NNNEEE, NNNEEN, NNENNE, NNENEN, NNENEE, NNEENN, NNEENE, NENNNE)
- (NENEN, NNNNEE, NNNENE, NNNEEE, NNNEEN, NNENNE, NNENEN, NNENEE, NNEENN, NNEENE, NENNNE, NENNEN, NENNEE)
- (NNNNEE, NNNENE, NNNEEE, NNNEEN, NNENNE, NNENEN, NNENEE, NNEENN, NNEENE, NENNNE, NENNEN, NENNEE, NENENN, NENENE)

- (NNNENE, NNNEEE, NNNEEN, NNENNE, NNENEN, NNENEE, NNEENN, NNEENE, NENNNE, NENNEN, NENNEE, NENENN, NENENE, NNNNEEE)
- (NNNEEE, NNNEEN, NNENNE, NNENEN, NNENEE, NNEENN, NNEENE, NENNNE, NENNEN, NENNEE, NENENN, NENENE NNNNEEE, NNNENEE)
- (NNNEEN, NNENNE, NNENEN, NNENEE, NNEENN, NNEENE, NENNNE, NENNEN, NENNEE, NENENN, NENENE, NNNNEEE NNNENEE, NNNEEEN)

⋮ (×10 TIMES)

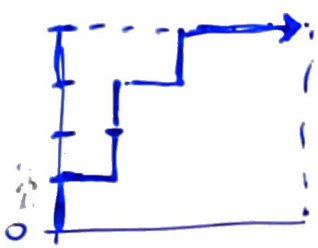
- (NNNNEEE, NNNENEE, NNNEEEN, NNNEENE, NNENNEE, NNENENE, NNENEEN, NNEENNE, NNEENEN, NENNNEE, NENNENE, NENNEEN, NENENNE, NENENEN)

⋮ (×13 TIMES)

- (NNNNEEEE, NNNENEEE, NNNEEENE, NNNEENEE, NNENNEEE, NNENENEE, NNENEENE, NNEENNEN, NNEENENE, NENNNEEE, NENNENEE, NENNEENE, NENENNEE, NENENENE)

NOW LET'S GO TO THE OPTIMIZED CODE.
A GOOD DATA-STRUCTURE TO REPRESENT THE PATHS IS AN ARRAY OF INTEGERS OF LENGTH N, WHERE EACH INTEGER REPRESENTS THE CUMULATIVE NUMBER OF EAST STEPS FOR THE i-th NORTH STEP, BEING i THE POSITION IN THE VECTOR. FOR EXAMPLE

$[0, 1, 1, 2]$ ENCODES THE PATH



SO AFTER FIRST STEP, THAT IS MANDATORY TOWARDS NORTH, WE HAVE ϕ EAST STEPS.
AFTER THE SECOND STEP TOWARDS NORTH WE DID ALSO 1 STEP EAST AND SO ON

CLEARLY $P[i] \leq i \quad i \in \{0, 1, \ldots, N-1\}$

AND THIS IMPLIES $P[0] \equiv 0$

IN ORDER TO PARALLELIZE THE CREATION OF THE PATHS WE NEED A PROCEDURE TO CREATE A VALID PATH P_{i+1} FROM A GIVEN VALID PATH P_i, AND WE NEED ALSO A PROCEDURE TO CREATE A VALID PATH P_k GIVEN AN INDEX k.

IN THAT WAY WE CAN CREATE SEVERAL STARTING PATHS FOR DIFFERENT VALUES OF k AND WE CAN GENERATE FROM THEM ALL OTHER MISSING PATHS.

FOR EXAMPLE IF WE WOULD LIKE TO CREATE ALL $C_4 = 14$ PATHS HAVING THE CAPABILITY TO RUN 4 PARALLEL PROCESSES, WE CAN CREATE 4 STARTING PATHS, LET'S SAY P_1, P_5, P_9, P_{13} AND GENERATE THE REMAINING AS FOLLOWS

$(P_1) \rightarrow P_2 \rightarrow P_3 \rightarrow P_4$

$(P_5) \rightarrow P_6 \rightarrow P_7 \rightarrow P_8$

$(P_9) \rightarrow P_{10} \rightarrow P_{11} \rightarrow P_{12}$

$(P_{13}) \rightarrow P_{14}$

HERE THE DESCRIPTION OF THE FUNCTION TO GENERATE A PATH FROM A GIVEN ONE: THE IDEA IS TO MOVE FROM THE RIGHTMOST ELEMENT OF THE GIVEN PATH AND SEE IF THE ARRAY ELEMENT CAN BE INCREASED BY 1. IF SO SET THAT VALUE TO ALL THE RIGHT ELEMENT TO THIS POSITION, OTHERWISE KEEP MOVING TO LEFT UNTIL THE END.

```
NEXTPATH (P)        ~> | ZERO-BASED ARRAY |
    INDEX ← LENGTH(P) - 1
    WHILE INDEX > 0
        IF P[INDEX] < INDEX
            P[INDEX] += 1
            CURRENT ← INDEX + 1
            WHILE CURRENT < LENGTH(P)
                P[CURRENT] ← P[INDEX]
                CURRENT += 1
            RETURN TRUE
        INDEX -= 1
    RETURN FALSE   <~ THIS HAPPEN ONLY ON THE PATH [0,1,2,3,4,....]
```

NOW BEFORE DESCRIBING THE ALGORITHM TO GENERATE A PATH FROM AN INDEX i LET'S INTRODUCE THE "CATALAN TRIANGLE"

$$C_{n,k} = \binom{n+k}{k} - \binom{n+k}{k-1}$$

NOTE $C_{n,n} \equiv$ CATALAN NUMBER C_n

HERE A PORTION OF THE TRIANGLE

n \ k	0	1	2	3	4
0	1				
1	1	1			
2	1	2	2		
3	1	3	5	5	
4	1	4	9	14	14
.....					

RULES

$C_{n,0} = 1 \quad \forall n \geq 0$

$C_{n,1} = n \quad \forall n \geq 1$

$C_{n+1,k} = C_{n+1,k-1} + C_{n,k}$

$C_{n+1,n+1} = C_{n+1,n}$

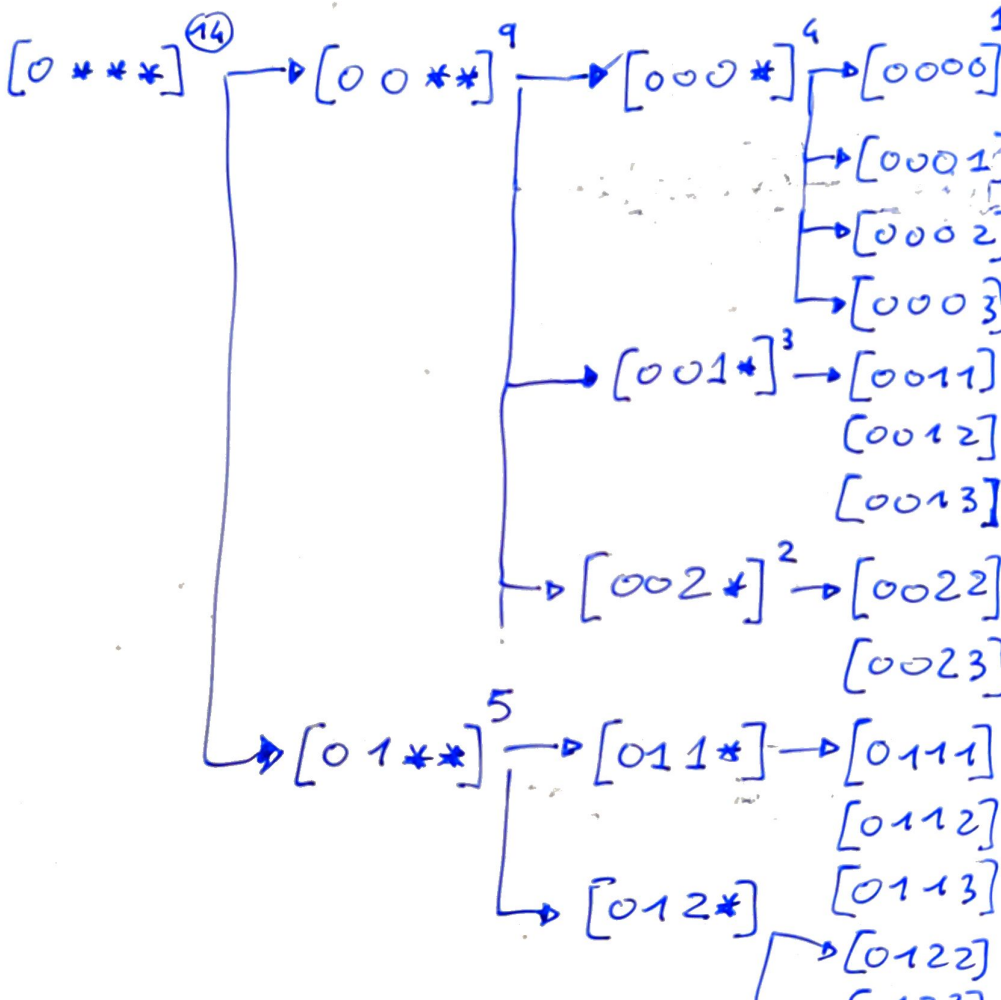
LET'S TAKE THE SUB-TRIANGLE IN RED

	0	1	2	3
0	1			
1	1	2		
2	1	3	5	
3	1	4	9	14

$\equiv CT[x][y]$

$x \geq 0 \quad y \geq 0$

THE $C_m = 14$ INDICATES THAT THERE ARE 14 PATHS CORRESPONDING TO 14 COMBINATIONS, ALL OF TYPE $[0, *, *, *]$ BEING * NOT SET. THESE 14 PATHS CAN BE SPLIT BY 9 OF THE TYPE $[0, 0, *, *]$ AND 5 OF THE TYPE $[0, 1, *, *]$ AND GOING IN THAT WAY WE CAN ENUMERATE AND BUILD ALL OF THEM:



SO GIVEN THE RED SUB-TRIANGLE AND AN INDEX i, IT IS POSSIBLE BY PROPERLY NAVIGATING THAT TRIANGLE TO BUILD THE COMBINATION.

HERE THE PSEUDOCODE

```
GENERATEPATH(CT, N, i)        # CT: THE RED CATALAN TRIANGLE
    PATH ← ZEROS(N)
    N -= 1
    p ← ∅
    s ← ∅
    PATH[p] ← s
    p += 1
    WHILE p ≤ N
        WHILE CT[N-s][N-p] < i
            i ← i - CT[N-s][N-p]
            s += 1
        P[p] ← s
        p += 1
    RETURN PATH
```