

PSEUDO CODE

TREE\_NODE

int VAL

TREE\_NODE\* LEFT, RIGHT

TREE\_NODE(int v)

VAL=v

VECTOR<VECTOR<TREENODE\*>> CORRECT\_PATHS(TREE\_NODE\* root, int X)

VECTOR<VECTOR<TREENODE\*>> PATHS

VECTOR<TREENODE\*> PATH

DFS(root, PATH, 0, X, PATHS)

RETURN PATHS

VOID DFS (TREE\_NODE\* NODE, VECTOR<TREENODE\*>& PATH, INT CURR\_SUM, TARGET\_SUM,

VECTOR<VECTOR<TREENODE\*>>& PATHS)

if NODE == NIL

RETURN

PATH. PUSH-BACK(NODE)

CURR-SUM += NODE->VAL

if (NODE->LEFT AND NODE->RIGHT == NIL AND CURR-SUM == TARGET-SUM)

PATHS. PUSH-BACK(PATH)

DFS(NODE->LEFT, PATH, CURR-SUM, TARGET-SUM, PATHS)

DFS(NODE->RIGHT, PATH, CURR-SUM, TARGET-SUM, PATHS)

PATH. POP-BACK()

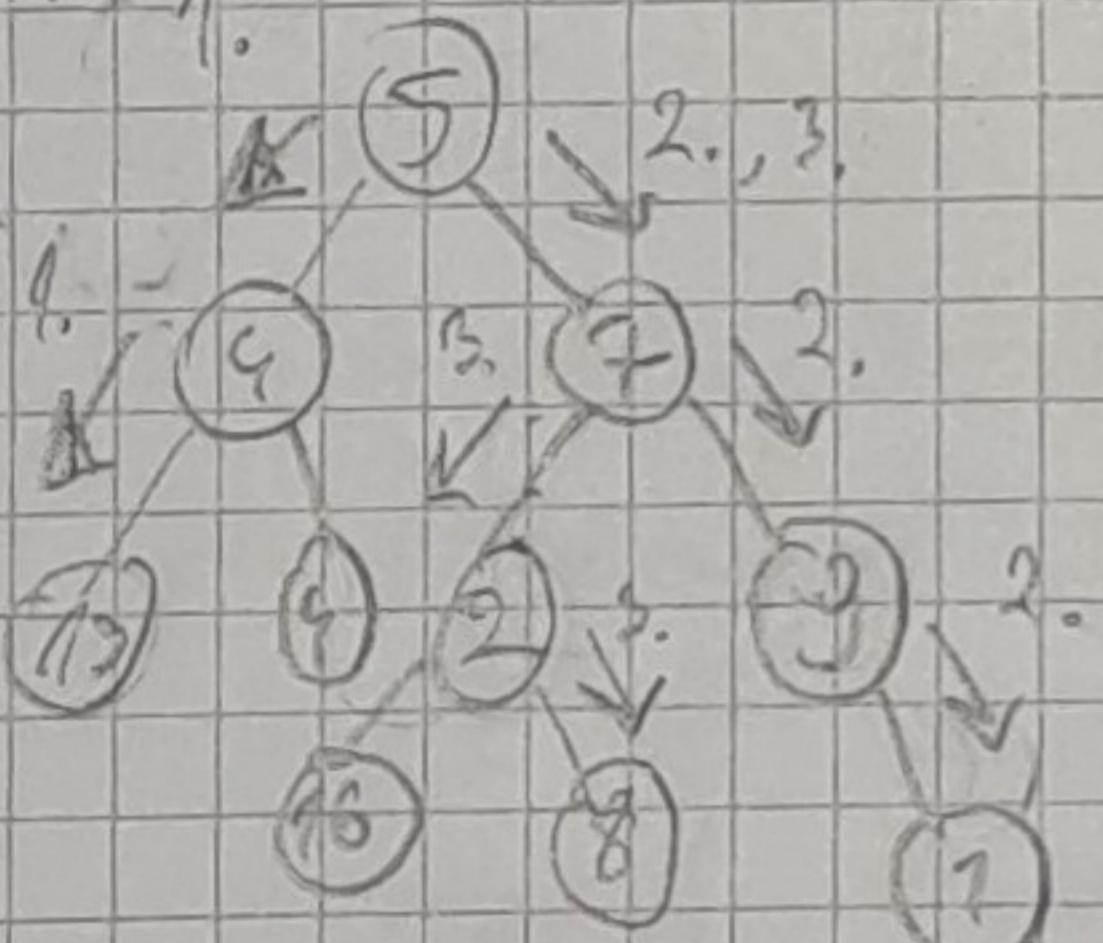
Algorytm mudi się otoże nie powie CORRECT\_PATHS mu  
możt o określonym numerze, on wtedy vector w którym, ją  
gdyż ma przedstawiać vector y były mu pushbackon  
możliwe bije mu problem i tyle numer dodatkowo TARGET

Npr. už te f-ji mix bit. <<8,9,5,6>, <3,2,11>>. To  
vise sijdi mode, a ne rano vrijednost množe.

Gloranu ulogu imo f-ju DFS. Onu rekušiv pređi  
kao mthod i pushuje u nudi mode, te očekuje  
slojevi množe bit i tentu novu jedinicu gi  
vrijednost. Tako se novi put koji odgovara i novi  
put pushuje u vector putova. Zatim rekušiv iden-  
tificiše liji; alesmu stvar, o kome bi se moglo  
da se radi i to da funkcija monača ovi pop-back()  
kao množe dodjeli množe. To nekošto je uobičajeno da bilo, množe  
se vrati i povezati sluge puteve (backtracking).

VSA ovaj algoritam je  $O(n^2)$ , gde je n preostalo  
broj množava u BST-u. Vrijedne potiske su DFS BST-a  
je  $O(m)$ , ali može biti da se običit. ob simbol put  
odgovarajućim množavima. Ovo daje  $O(nm^2)$ .

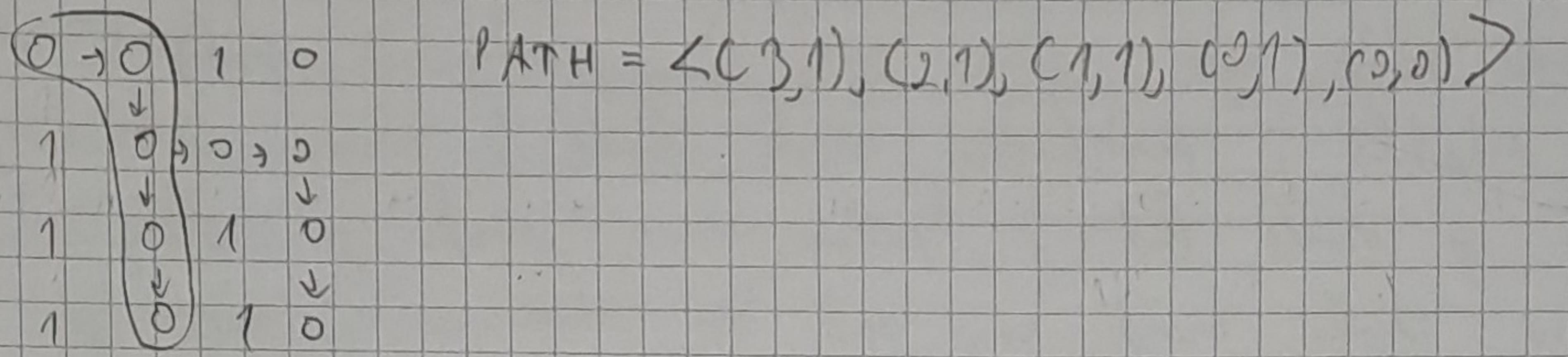
Primer 1.  $\sum = 22$



Rezultujući podnizovi jedinica su, a  
stek je odgovarajući krajnji put.

$$\text{PATHS} = \langle \langle 5, 9, 13 \rangle, \langle 5, 7, 3, 12, 14, 2, 1, 8 \rangle \rangle$$

2.  
Upr. matice  $5 \times 5$



PSEUDOKOD ZA FIGU BOJA VRAĆI MAJKRAĆI PUT U MATRICI

CONST INT MAX = 1000;

INT MAZE[MAX][MAX] → LABIRINT U MAINU ŠAM ( PRAVIMO )

INT DIST[MAX][MAX];

INT dx[2] = {1, 0} → DIREKCIJE

INT dy[2] = {0, 1} → DIREKCIJE

INT m → DIMENZIJA LABIRINTA

BOOL IS\_VALID(INT X, INT Y) → PROGRAMA JEGI (X, Y) UNUTR  
LABIRINTA I JE LI NUE 310

RETURN (X >= 0 AND X < m AND Y >= 0 AND Y < m AND MAZE[X][Y] == 0)

VECTOR<PAIR<INT, INT>> BFSC INT SX, INT SY, INT GX, INT EY)

QUEUE<PAIR<INT, INT>> Q → REGA ZA BST

VECTOR<PAIR<INT, INT>> PATH → RETURN VRJEDNOST

Q. PUSH({SX, SY})

DIST[SX][SY] = 0 → NAJMANJA UDALJENOST DO POCETNE TOČKE DO SAME  
WHILE (!Q.EMPTY()) SEBE JE 0

INT X = Q.FRONT.FIRST

INT Y = Q.FRONT.SECOND

Q. POP()

IF(X == BX AND Y == EY)

BREAK → KRAJNA TOČKA

FORC i = 0; i < 2; i++) → SVAKO POMIŠLJAJE DOŠAO UZ DESNO

INT MX = X + dx[i]

INT MY = Y + dy[i] NOVE TOČKE

IF (IS\_VALID(mx, my) AND DIST[mx][my] == INT\_MAX)

// Ako je novi točki VALID (nije pogrešna)

DIST[mx][my] = DIST[X][Y] + 1 → NAJMANJA UDALJENOST

Q. PUSH({mx, my}) DO NOVE TOČKE

// KONSTRUKTOR PUT DO KRAJNE TOČKE DO POCETNE

INT X = BX, Y = EY

PATH.PUSH\_NODE({X, Y})

WHILEC X != SX || Y != SY → DOK NE DODEMO DO POCETNE TOČKE

// POGLEDAJ OKOLNE TOČKE (ODMOSR) IJU S NAJMANJOM UDALJENOSTI

FOR(i = 0; i < 2; i++)

INT mx = X + dx[i]

INT my = Y + dy[i]

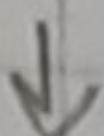
IF (IS\_VALID(mx, my) AND DIST[mx][my] ≤ DIST[X][Y] - 1)

X = mx, Y = my

PATH.PUSH\_NODE({X, Y})

BREAK

1 ROTURNI REVERSE (PATH)



PRVO JE INSERTANA KRAJNA TOČKA  
(BACK TRACKAMO)

U MAJMU SE UVEĆI M, ELEMENTI LABIRINTA,  
SU ELEMENTI DISTRA SE STAVLJAJU NA INT\_MAX  
BFS SE POZIVA BFS(0, 0, m-1, m-1)

Nekon išto me učivo dimenzije labirinta i novi bekint,  
koji je globalna varijabla prvo se BFS koji vrati red  
za pretraživanje u kojem do sada došlo je u svim  
je reda i f je pozivom susjedne točke koji je mogu  
stavi poziciju u desno ili lijevo. Ako je susjedna  
točka VAL, tada je učinjeni neophodni udaljenost sva  
te točke i sljedeći je u red. Novi red obnovi se  
krajnje točke, f je postavljena, tako da može putu  
Do sada došlo je da posetio poziciju učinkovit  
u posjetu krajnjim u dist tablici. Dimen  
možnostima učinkoviti. U vrijednost dodaji (u metrički  
merni jedinicu) algoritam preduzim sada  $\Rightarrow O(N \cdot M)$   
N, M dim. MATRICE.