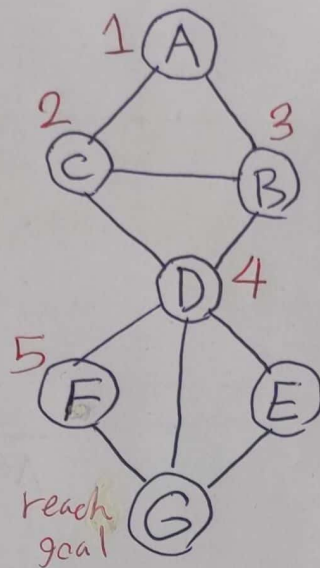
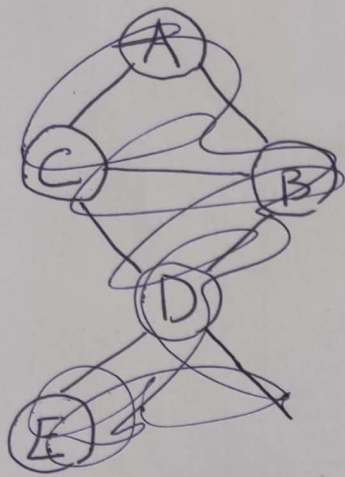


۹۹۵۲۱۲۷۱

فرزان رحمانی (بخش تئوری) HW1-AI



سؤال ۱ -

بافرض اینکه گره‌های دیده شده را در آرایه
 Visited ذخیره کنیم و دوباره آنها را
 Visit نکنیم.

queue: IA → (I)IBC → (II)IDB → (III)ID → ~~ABC~~ (IV)IEF → (V)IGE (VI)

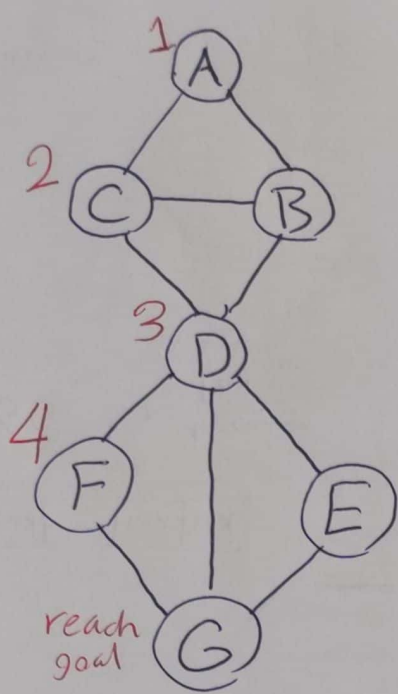
به هدف رسیدیم پس الگوریتم stop می شود.

① BFS
 expand a shallowest Node First

FIFO stack
 queue
 صف

ترتیب گره‌های
 expand شده: A, C, B, D, F

مسیر پیراشده: A → C → D → F → G



ادامه سوال 1
* گره های تکراری را visit نمی کنیم.

stack:

↑
A

 (I)

↑
C
B

 (II)

↑
D
B

 (III)

↑
F
E
B

 (IV)

→ به هدف رسیدیم پس الگوریتم stop می شود.

G
E
B

 (V)

② DFS

expand a deepest node first

LIFO stack

ترتیب گره های
expand شده: A, C, D, F

مسیر پیراشده: $A \rightarrow C \rightarrow D \rightarrow F \rightarrow G$

Iterative Deepening get DFS's space

advantage with BFS's time/shallow-solution advantage

سوال ۲

این روش ترکیب DFS و BFS هست و مزیت حافظه DFS و زمان BFS را دارد ولی کار اضافه از order $O(b^{s-1})$ دارد و یکم redundant هست ولی چون بخش عمده

extra work $s = \text{smallest depth of solution}$ کار در آخرین مرحله search رخ می دهد خیلی تأثیر بدی ندارد.

(2)

Frontier: stack

Completeness: yes

Optimal: No (except when all edges costs same)

Time: $O(b^s)$

b = branching factor

Space: $O(bs)$

s = smallest depth of solution

Iterative
Deepening
Properties

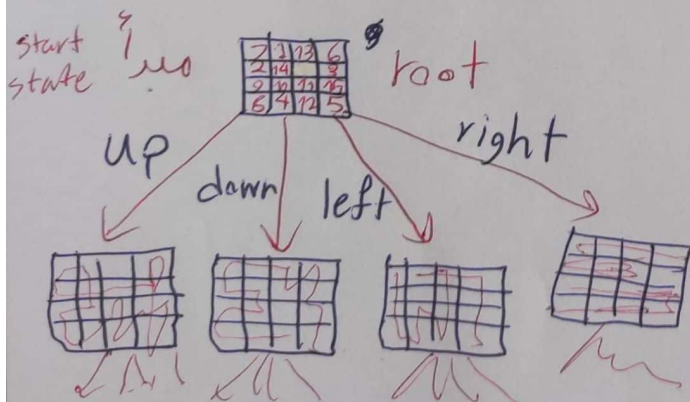
ادامه سوال ۲

این الگوریتم به علت مزایای حافظه DFS و زمان BFS کاربردهای زیادی در AI دارد مثل کاربردهای در بازی ها مانند Fifteen Puzzle و Chess programs و همچنین طور زمان هایی که عمق جواب را نمی دانیم و state space بزرگی داریم.

سوال ۳. با هر دو روش BFS و DFS می توان این پازل را حل کرد.

به این صورت که هر یک از حالت های $state$ که خانه ها قرار گرفته اند را یک node گراف در نظر بگیریم و هر یک از اعمال $up, down, left, right$ برای خانه خالی یک edge گراف در نظر بگیریم. حال حالت ابتدایی بازی را به عنوان root در tree search $start\ state$

در نظر گرفته و از آن شروع می کنیم و هر یک از الگوریتم های BFS و DFS را اجرا می کنیم تا به goal state برسیم و سپس یال های این مسیر $actions\ sequence$

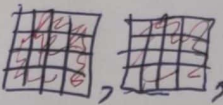


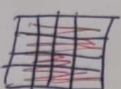
را برای بردن بازی به ما می دهند. زمانی که به این node (goal state) رسیدیم الگوریتم را stop می کنیم.

ضمناً از دیدن node های تکراری جلوگیری می کنیم تا در infinite loop قرار نگیریم.

goal state

3

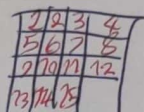
state space: حالت‌های خانه‌ها  → nodes

initial state: حالت اولیه خانه  → root

Action Cost:

+1 per step

Actions: up, down, left, right → edges

Goal test(state): خانه‌های مرتب شده 

در این مسئله { $\text{branching factor} = 4 = b$
 $m = \text{max depth of search tree}$
 $S = \text{smallest depth of solution}$

اگر از دیدن node های تکراری جلوگیری کنیم m و S محدود finite می شوند.

DFS: Time: $O(b^m) = O(4^m)$

stack

Space: $O(bm) = O(4m)$

جواب لزوماً optimal نیست.
 complete هست چون محدود است فضای حالت و از دیدن node های تکراری جلوگیری کردیم.

BFS: Time: $O(b^S) = O(4^S)$

queue

Space: $O(b^S) = O(4^S)$

جواب optimal هست چون cost همه یال‌ها یکی می باشد.

complete هست چون جواب دارد و محدود است.