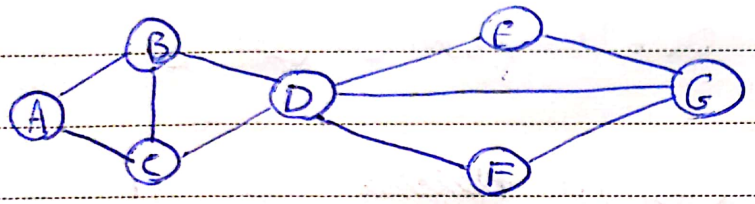

BFS: A → B, C → D → E, G, F ABDG : مسیر

عمق = 0 عمق = 1 عمق = 2 عمق = 3

DFS: A → B → D → E → G → F → C ABDEG : مسیر

SALEH

2- روشی برای search کردن در گراف است که از مزایای iterative Deepening

DFS و BFS به طور همزمان استفاده می‌کند

مشکل روش DFS این بود که بهینه نبود و ممکن بود با وجود نودهای کم عمق

تر یک یا نودهای عمق دور را انتخاب کند.

BFS جواب بهینه را پیدا می‌کند اما به حافظه زیادی برای ذخیره نودها نیاز

دارد.

در Iterative Deepening ، DFS را برای عمق های متفاوت اجرا می کنیم.

ابتدا عمق 1، سپس 2 و 3 و 4 و ... تا زمانی که جواب، به این ترتیب هم جواب های بهینه (با کم ترین عمق) را پیدا می کنیم هم نیازی به حافظه زیاد نداریم.

اردر زمانی اجرای این الگوریتم $O(b^d)$ است که از همان اردر DFS و BFS است.

در AI از این روش برای جست و جو در state graph ها استفاده می شود تا بتواند جواب بهینه را با حافظه کم پیدا کند و به ازای state های بزرگ

3 - ه برای حل این سوال تعداد state ها و action ها را بررسی می کنیم

جدول بازی 4*4 است و یک فضای خالی دارد

تعداد state ها برابر است با :

$$16 \times 15! = 16!$$

تعداد حالت های فضای خالی

در هر اکشن می توانیم فضای خالی را به 4 جهت حرکت دهیم

مگر در کناره های جدول .

BFS و DFS هر دو از O(V+E) هستند بنابراین از O(16!) است.
بنابراین اگر با محدودیت زمان مواجه نباشیم می توان از BFS و DFS استفاده کرد اما در عمل با هیچ کامپیوتری نمی توان با این روش جواب را پیدا کرد.

اگر تعداد حرکت های لازم کم باشد، با BFS می توان سریع به جواب رسید زیرا راه حل بهینه ای را پیدا می کند