**گزارش پروژه اول**

**اشکان شکیبا (9931030)**

**سوال صفر**

کلاس SearchProblem یک کلاس انتزاعی (abstract) است که ساختار کلی یک مسئله جست‌و‌جو را تعیین می‌کند.

متد getStartState وضعیت آغازین مسئله را بازگردانی می‌کند.

متد isGoalState یک ورودی state می‌گیرد و در صورتی که وضعیت هدف باشد، True بازگردانی می‌کند.

متد getSuccessors نیز یک ورودی state می‌گیرد و لیستی از مجموعه سه‌تایی successor، action و step cost بازگردانی می‌کند که شامل successorهایی بر وضعیت فعلی به همراه action مورد نیاز برای رسیدن به آن و هزینه انجام آن است.

متد getCostOfActions یک ورودی actions می‌گیرد که شامل لیستی از کنش‌هاست و مجموعه هزینه انجام دنباله آنها را بازگردانی می‌کند.

کاربرد کلاس‌های فایل game.py:

Agent: پیاده‌سازی عامل‌ها و بازگردانی کنش‌های آن‌ها

Directions: ذخیره جهت‌های جغرافیایی و جهت‌های نسبی

Configuration: نگه‌داری مختصات موقعیت و جهت حرکت عامل

AgentState: نگه‌داری وضعیت یک عامل، شامل موقعیت، جهت و سرعت حرکت و ...

Grid: نگه‌داری زمین بازی در قالب یک آرایه دو بعدی

سوال یک

در الگوریتم IDS، بخشی از گراف که تا قبل از عمق تعیین‌شده‌ای قرار دارد انتخاب شده و در آن بخش به جست‌وجو با DFS می‌پردازیم. به بیان دیگر الگوریتم DFS از عمق تعیین‌شده پایین‌تر نمی‌رود و در صورتی که در محدوده انتخاب شده به هدف نرسیم، مقدار عمق تعیین‌شده به مرور افزایش می‌یابد.

بنابراین می‌توان گفت برای تغییر DFS به IDS باید در چند مرحله و تا زمان رسیدن به هدف، عمق مشخصی تعیین کرد که الگوریتم DFS در هر مرحله از عمق مشخص‌شده در آن پایین‌تر نرود.

شبه کد الگوریتم IDS:

**function** IDS(root) **is**

**for** depth **from** 0 **to** ∞ **do**

found, remaining ← DLS(root, depth)

**if** found ≠ **null** **then**

**return** found

**else** **if** not remaining **then**

**return** **null**

**function** DLS(node, depth) **is**

**if** depth = 0 **then**

**if** node is a goal **then**

**return** (node, **true**)

**else**

**return** (**null**, **true**) *(Not found, but may have children)*

**else** **if** depth > 0 **then**

any\_remaining ← **false**

**foreach** child **of** node **do**

found, remaining ← DLS(child, depth−1)

**if** found ≠ **null** **then**

**return** (found, **true**)

**if** remaining **then**

any\_remaining ← **true**

*(At least one node found at depth, let IDS deepen)*

**return** (**null**, any\_remaining)