**איך בנוי הפרוייקט ?**

התשתית לכל התרגיל מבוססת על graph\_problem\_interface:

* class GraphProblemError(Exception): חריגה המודיעה על שגיאה בגרף
* class GraphProblemState(abc.ABC): מחלקה אבסטרקטית עבור מצב במרחב המצבים שלנו:

מתודות :

* "eq"- מגדירה מתי שני מצבים נחשבים שווים זה לזה. אסור להשוות float עם ==.
* "hash"- מתודה הנותנת מספר ייחודי לכל מצב (באמצעות הפעלת מתודה כזו למשל בודקים בקבוצה האם המצב שייך כבר לקבוצה). קריטי שלשני אובייקטים המייצגים את אותו מצב יהיה אותו האש.

נהוג להשתמש בtuple של כמה שדות מהאובייקט כדי ליצור את ההאש. לא להשתמש בשדה מסוג float.

* "str"- מגדירה איזה מחרוזת תודפס כאשר למשל נעשה print על אובייקט מסוג זה.\
* מחלקת ExtendedCost- מחלקה אבסטרקטית על מנת לתאר ממשק של טיפוס למחיר. מחלקה כזו נחוצה כאשר לבעיה יש מס' פונקציות המשמות למחיר, כשכל אחת מהן מצטברת עצמאית במהלך החיפוש.

"g\_cost"- סקלר מסוג float אשר יעבור אופטימזציה במהלך אלגוריתם החיפוש. הg\_cost יכול להיות למשל כל אחת מהפונקציות המצטברות או פונקציה של הפונקציות האלה.

מתודות:

* get\_g\_cost – מחזירה float
* \_\_add\_\_- מקבלת את האובייקט עצמו וother שהוא מסוג ExtendedCost ומחזירה טיפוס מסוג ExtendedCost שהוא החיבור של שניהם.
* Cost = Union[float, ExtendedCost] – טיפוס שיכול להיות אחד מ2 הסוגים: float או ExtenedCost.
* class OperatorResult(NamedTuple)- מבנה הנתונים namedtuple מאפשר לייצר טיופל עם שמות לשדות. זוהי מחלקה בעלת 3 שדות, שתחזיר את התוצאה של אופרטור כטיופל המכיל את המצב כתוצאה מהפעלת האופרטור , את המחיר של האופרטור ( שיהיה מסוג float או ExtenedCost ואופציונאלית תחזיר operator name גם (או שיהיה none ).
* Class StatesPathNode: מחלקה היוצרת טיופל עם שמות לשדות המכילה את השדות הבאים : מצב, את המחיר של האופרטור האחרון שהופעל, את העלות המצטברת של cost, את העלות המצטברת של g\_cost, ואופציונאלית את שם האופרטור האחרון שהופעל.
* Class GraphProblem: מחלקה אבסטרקטית שמייצגת את מרחב המצבים עצמו.

שדות: שדה name הוא שדה סטטי משותף לכל האובייקטים המכיל מחרוזת המכילה את שם הבעיה.

מתודות :

\_\_init\_\_: פונקציית בנאי המגדירה שדה פרטי לאובייקט- המצב ההתחלתי של הבעיה.

Expand\_state\_with\_costs: מתודה אבסטרקטית המקבלת מצב ומחזירה את כל המצבים העוקבים שלו

(succ(s . זהו איטרטור (כלומר אמור לממש את הפונקציה next, או לחילופין משתמש בyield). הוא מבצע איטרציה על המצבים העוקבים וכל פעם מחזיר מצב אחד עוקב באמצעות yield. לכל מצב מוחזר אובייקט מסוג operatorResult.

Is\_goal: מתודה אבסטרקטית המקבלת מצב ומחזירה האם הוא מצב מטרה.

Get\_zero\_cost : מתודה המחזירה אובייקט cost המייצג עלות 0. כמשתמשים בטיפוס של מחיר מורחב צריך לממש פונקציה זו במחלקה היורשת מחדש.

* מחלקת GraphProblemStatesPath(Tuple[StatesPathNode])- מייצגת מסלול של מצבים בגרף. מומשה על מנת לממש את המתודה str להדפסה. היא פשוט Tuple של StatesPathNode.
* מחלקת SearchNode: אובייקט במחלקה זו צומת הנוצרה כתוצאה מאלגוריתם חיפוש.

שדות באובייקט מסוג זה :

* GraphProblemState- המצב שאותו אובייקט מייצג
* צומת אבא מסוג searchNode
* מחיר (שווה למחיר של האופרטור ועוד המחיר של האבא)
* המחיר של האופרטור שבוצע כדי להגיע לצומת הזו
* הf -score של node- expanding priority