**זיווג:**

**באופן כללי:**

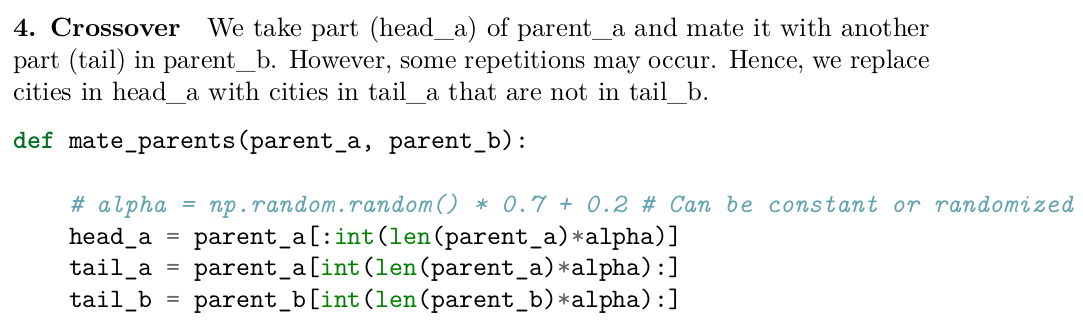
בפונקציה mate\_parents מתבצע זיווג לזוג פתרונות (הורים) מהצורה למשל 'ABC..' (אותיות כמספר הערים הנתונות) ויצירת פיתרון חדש המורכב משניהם שימשיך לדור הבא.

צורת הזיווג היא הגדרת "ראש" ו"זנב" לזוג הפתרונות לפי משתנה alpha (מספר בין 0 ל1).

הרציונל הוא ליצור פיתרון חדש מה"ראש" של פיתרון a ומהזנב של פיתרון b.

אך במקרה ויש לנו עיר (אות באנגלית) הנמצאת גם כאן וגם כאן, עלינו למצוא מנגנון שבמידה ואות מסויימת מה"ראש" של a נמצאת ב"זנב" של b **אז** תלקח אות מהזנב של a במקום (שם בהכרח האות אינה אותו הדבר.

המימוש בקוד:

****

**נתון:**

Parent a = “ABCD”

Parent b = “ACBD”

Alpha = 0.5

נקבל:

Head\_a = “AB”

Tail\_a = “CD”

Tail\_b = “BD”

כלומר האות B מופיעה גם בTail\_b וגם Head\_a.

לכן נשתמש במשתנה mapping בצורה הבאה:

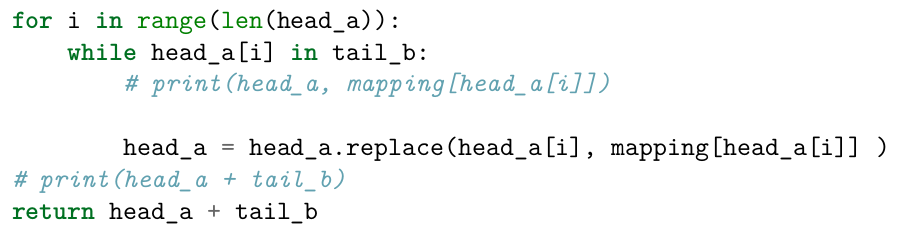


נקבל:

Mapping = {“B” : “C” , “D”: “D”}

משתנה שממפה את האותיות בזנב של b לאותיות בזנב של a

ולבסוף:



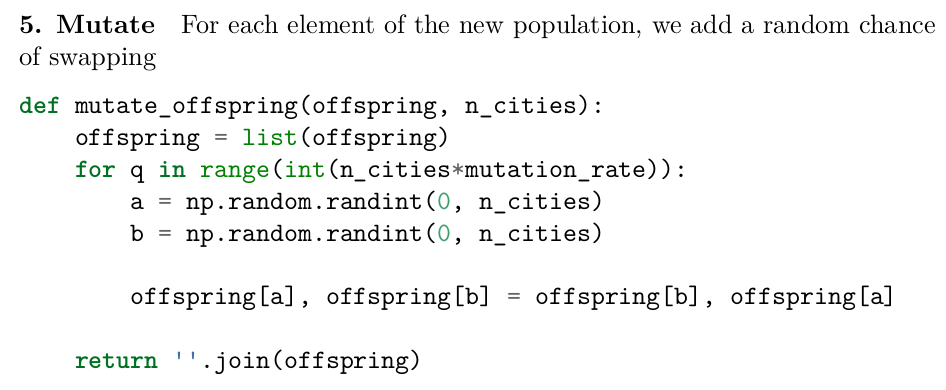
בלולאה בגודל head\_a : במידה והאות ה-i מופיע בtail\_b ניקח מהמילון mapping את המתאימה מtail\_a. כלומר במקרה שלנו ניתקל באות "B" המופיע כפול וניקח את ["mapping["B שזה בדיוק "C" ונחליף בhead\_a ונקבל בחיבור הסופי בreturn את "ACBD" שבמקרה הספציפי הזה יוצא אותו הדבר כמו parent\_b.

**מוטציה:**

**באופן כללי:**

הפונקציה mutate\_offspring לוקחת פיתרון מהסוג שהוזכר קודם וגורמת לשינוי אקראי בו על ידי החלפה של עיר אחת באחרת בתוך המסלול המאפיין אותו.

בתרגיל זה עבור כל פיתרון שנומצר לאחר הזיווג (צאצא) מתבצעת מוטציה, כאשר מספר ההחלפות שמתבצע עבור כל פיתרון מושפע ממשתנה mutation\_rate. ככל שיותר גדול -> יותר החלפות -> יותר גיוון. קטן יותר -> פחות החלפות -> שימור התכונות של הפיתרון הקיים.



למשל נתון:

n\_cities = 10 (הפיתרונות מורכבים מ10 אותיות באנגלית המייצגים מסלול אפשרי)

ן-יחס המוטציה - mutate\_rate = 0.2

נקבל: שעבור כל צאצא העובר מוטציה, יתבצעו 2 החלפות, ובכל החלפה כזאת ילקחו 2 ערים (a , b ) בצורה רנדומלית מתוך המסלול ויוחלפו ביניהן במיקום. ויווצר פיתרון חדש.