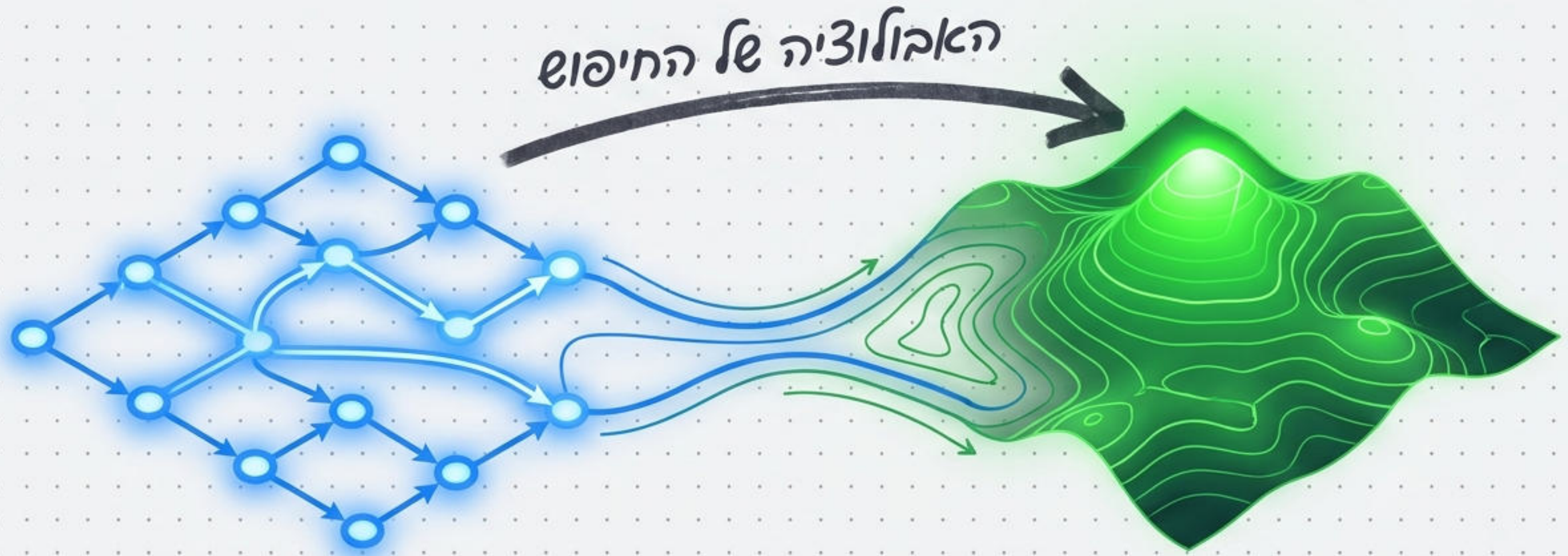
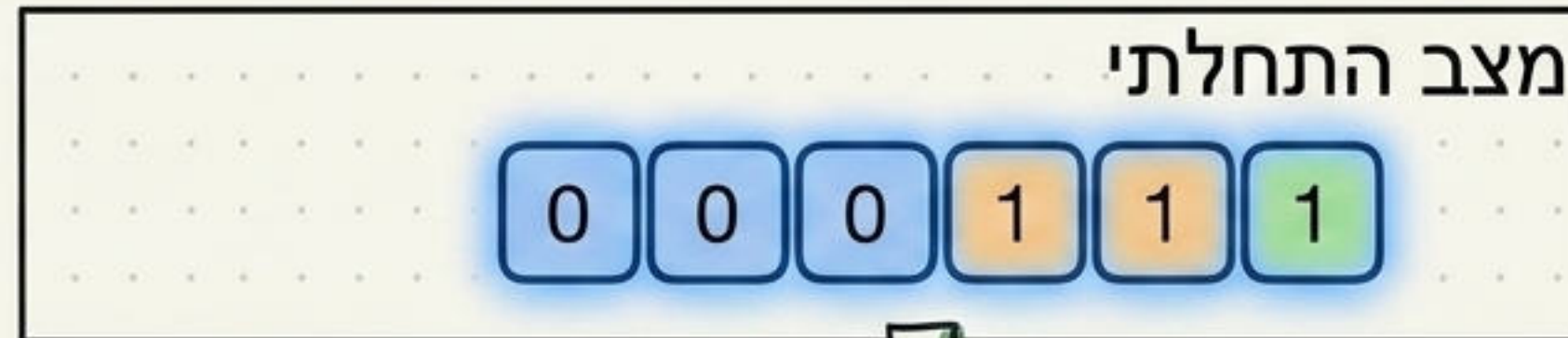
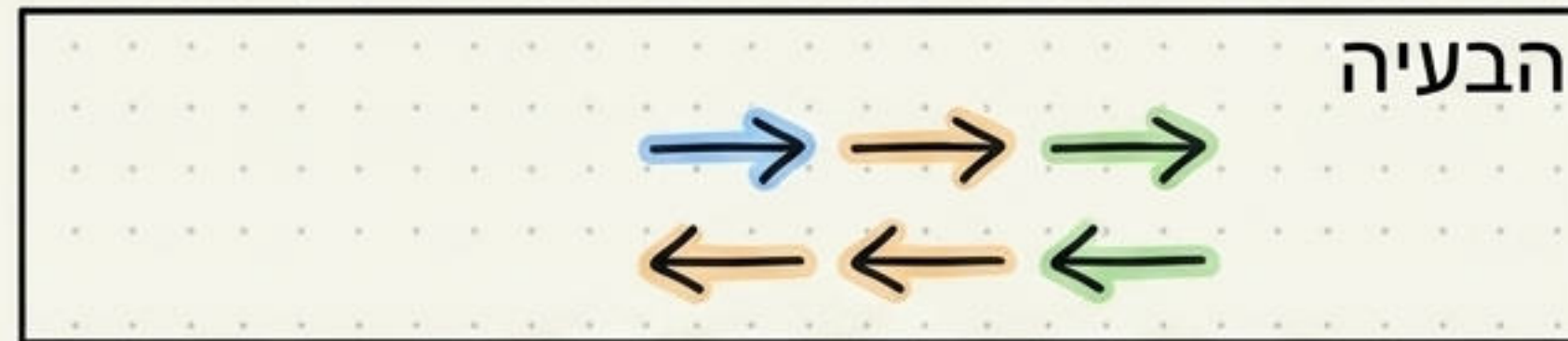


מבוא לחיפוש בבינה מלאכותית

מחיפוש קלאסי במרחב מצבים לחיפוש מקומי ואופטימיזציה



אנטומיה של בעיית חיפוש: מרחב המצבים

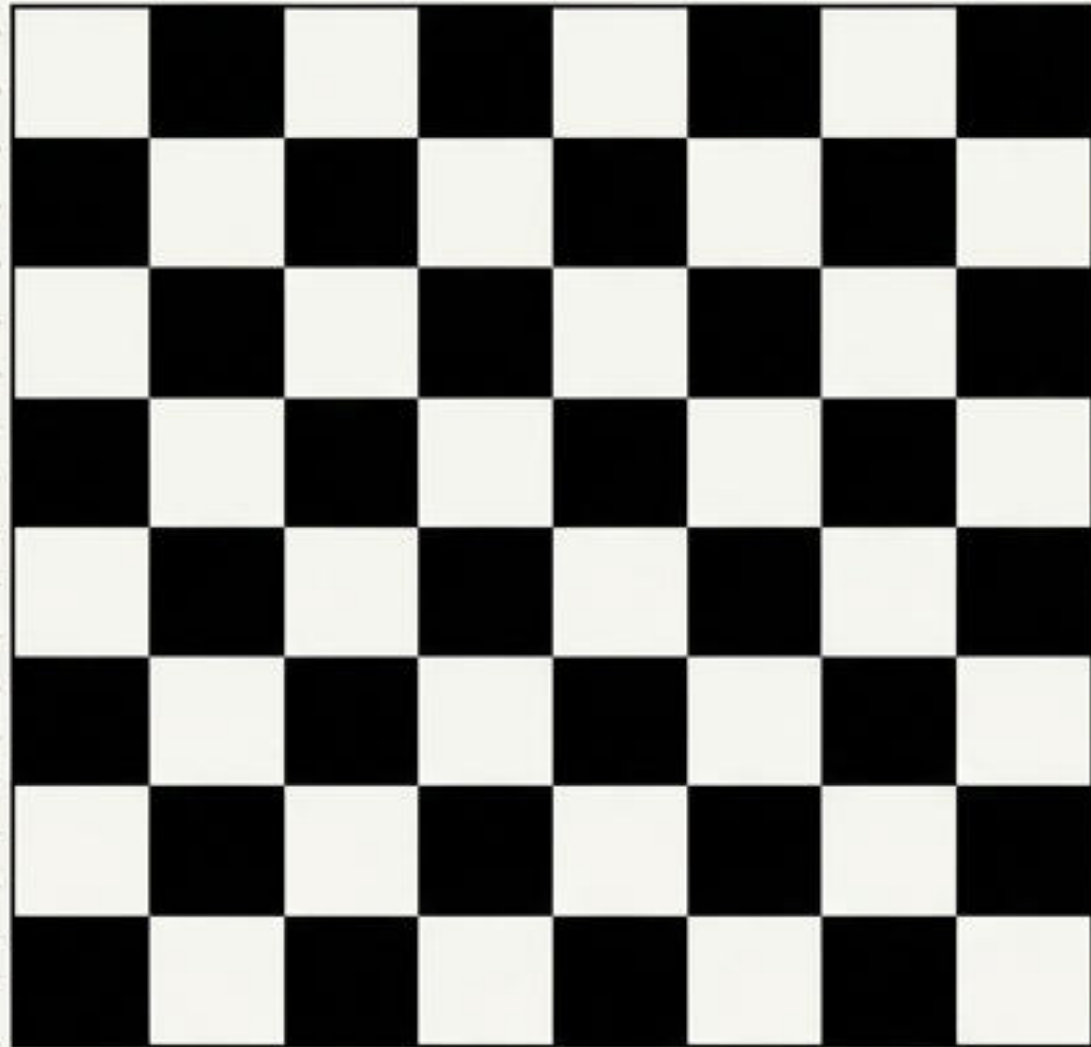


טיפ לסטודנט

פעולה: הפיכת שני
ביטים סמוכים
מקדם סידורי:
 $b = 5$

חיפוש מתחיל בייצוג נכון. מרחב המצבים מגדיר מאין מתחילים, לאן מגיעים, ואילו צעדים מותרים.

קללת המימדיות: כשהחיפוש יוצא משליטה



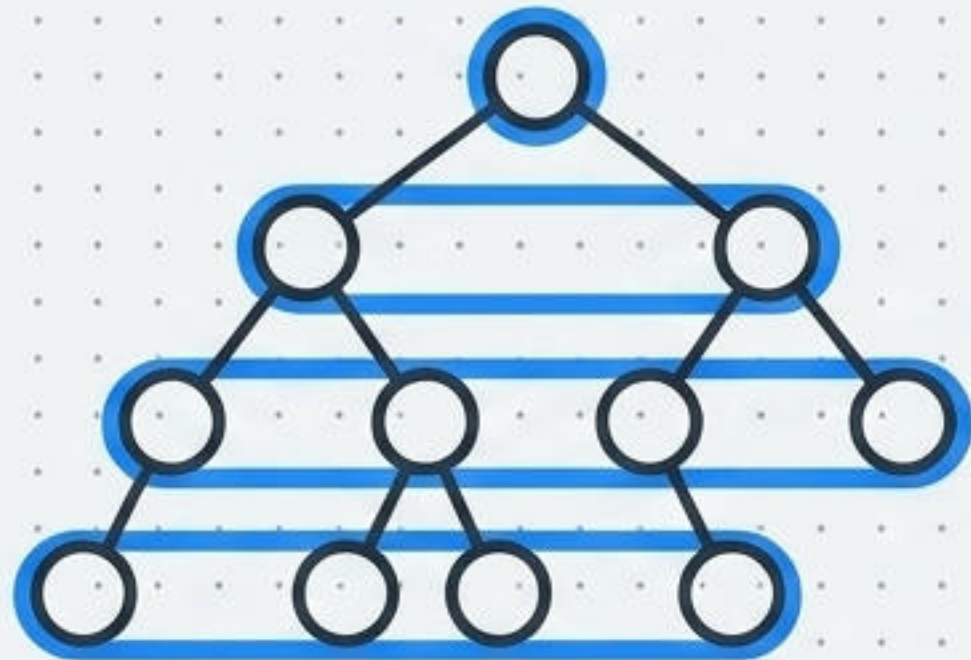
- גודל הלוח: 64 משבצות
- כל משבצת: שחור או לבן
- סך כל המצבים האפשריים: 2^{64}



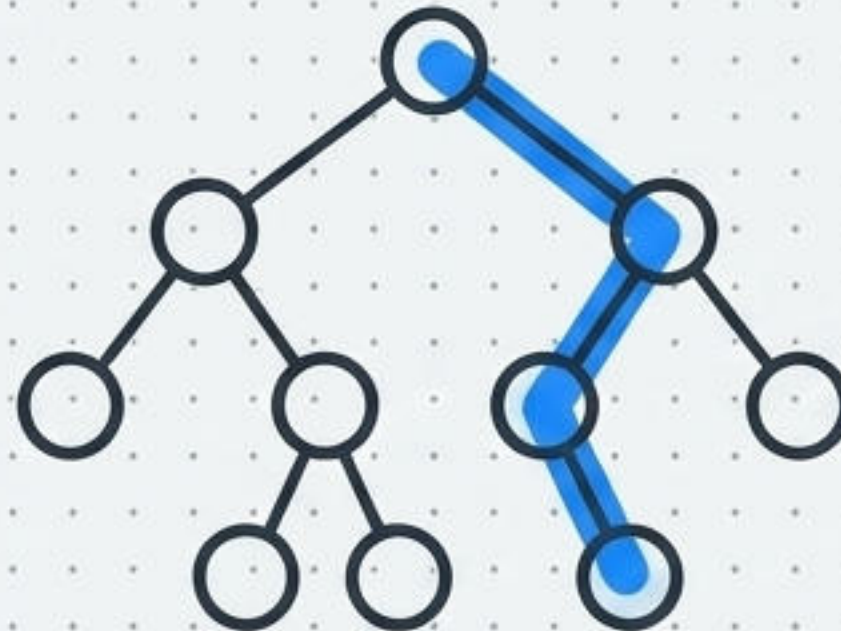
חיפוש עיוור ייכשל כאן. אנחנו חייבים להעריך
מראש איילו צעדים מקרבים אותנו למטרה.

חיפוש עיור: השוואת אסטרטגיות

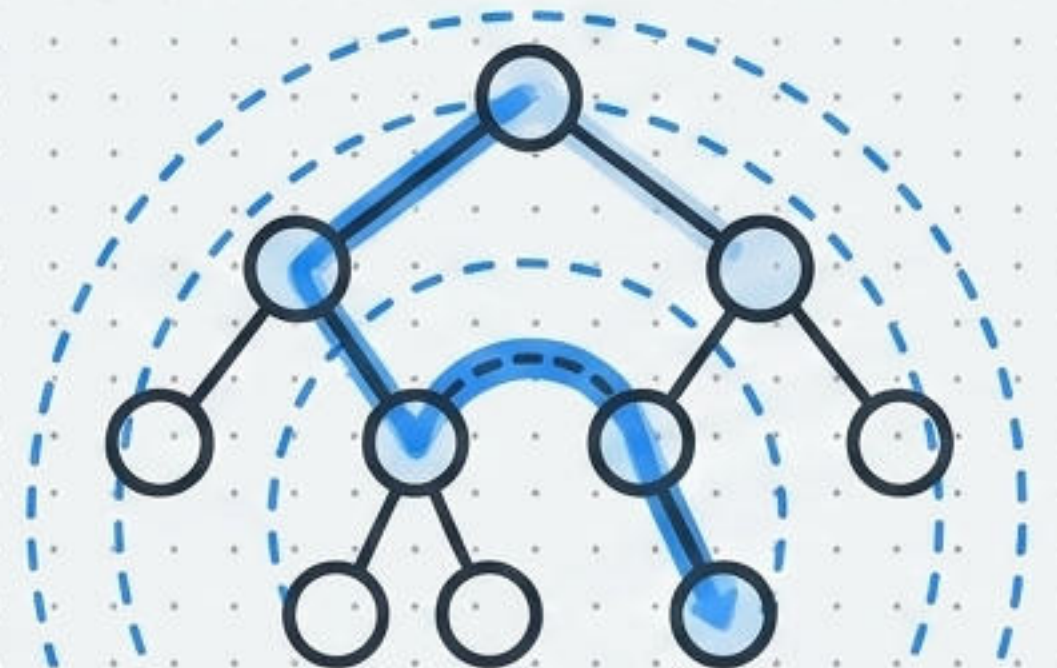
BFS



DFS



IDS

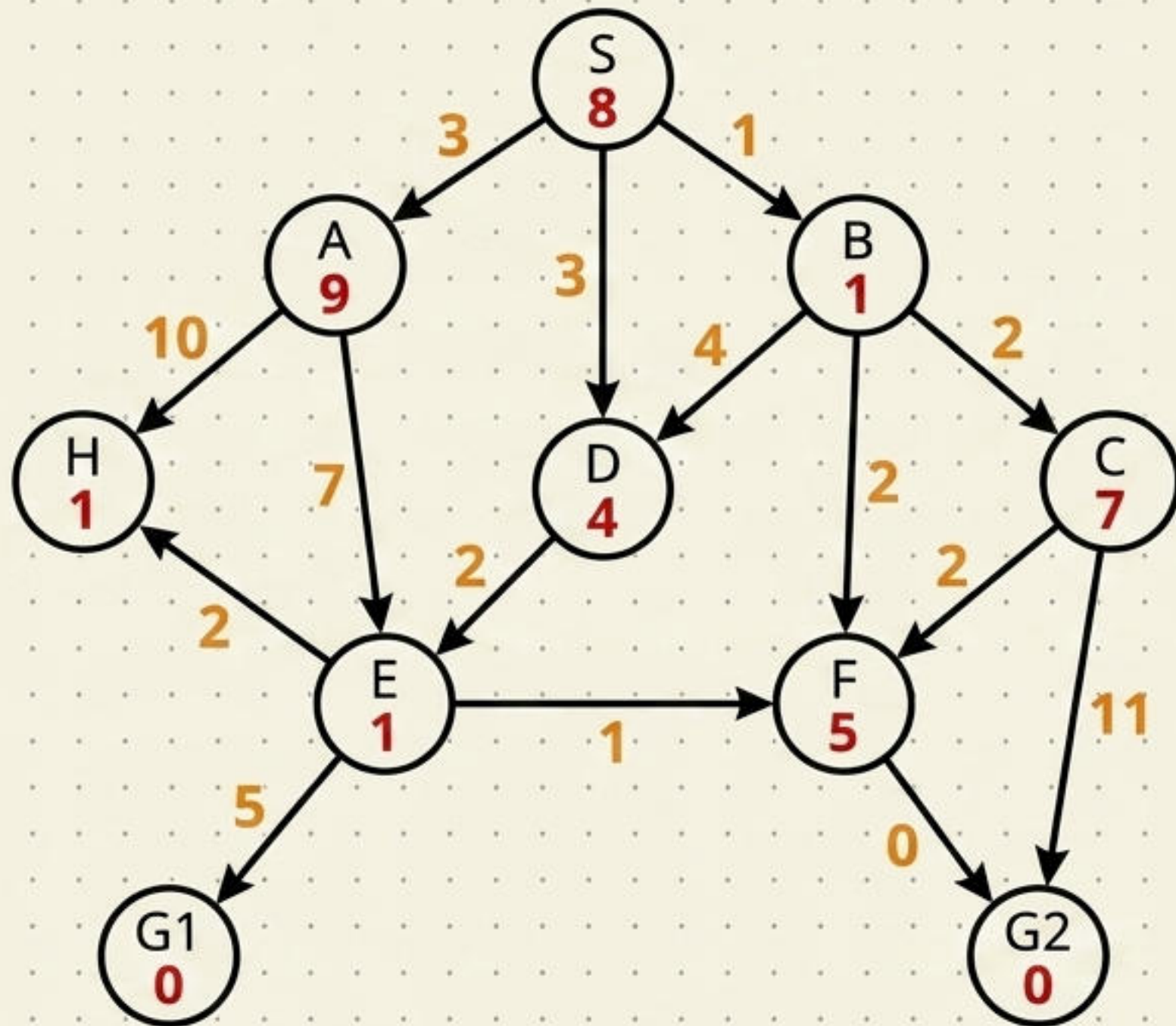


מתי IDS קורס?

במבנה של שרשרת, DFS ירוץ בזמן $O(n)$. לעומתו, IDS יבצע חישוב מחדש לכל עומק וידרוש $O(n^2)$.



חיפוש מושכל: הוספת היגיון לחיפוש



$$f(n) = g(n) + h(n)$$

עלות הדרך עד כה + הערכת המרחק שנוותר

קבילה (Admissible)

$$h(n) \leq h^*(n)$$

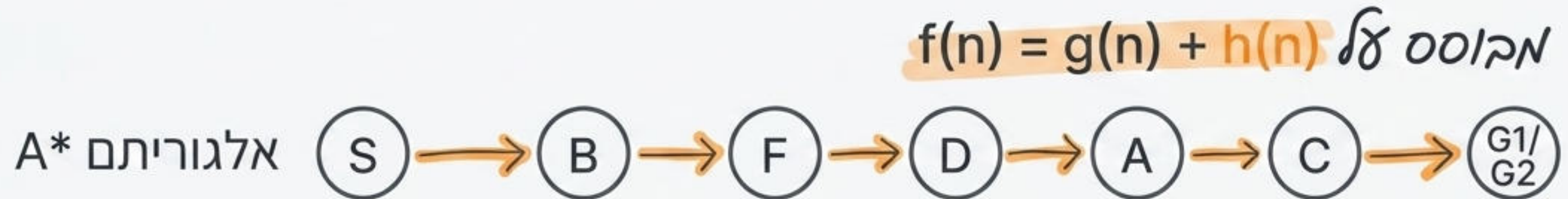
— לעולם אינה מגזימה בעלות.

עקבית (Consistent)

$$h(n) \leq c(n, a, n') + h(n')$$

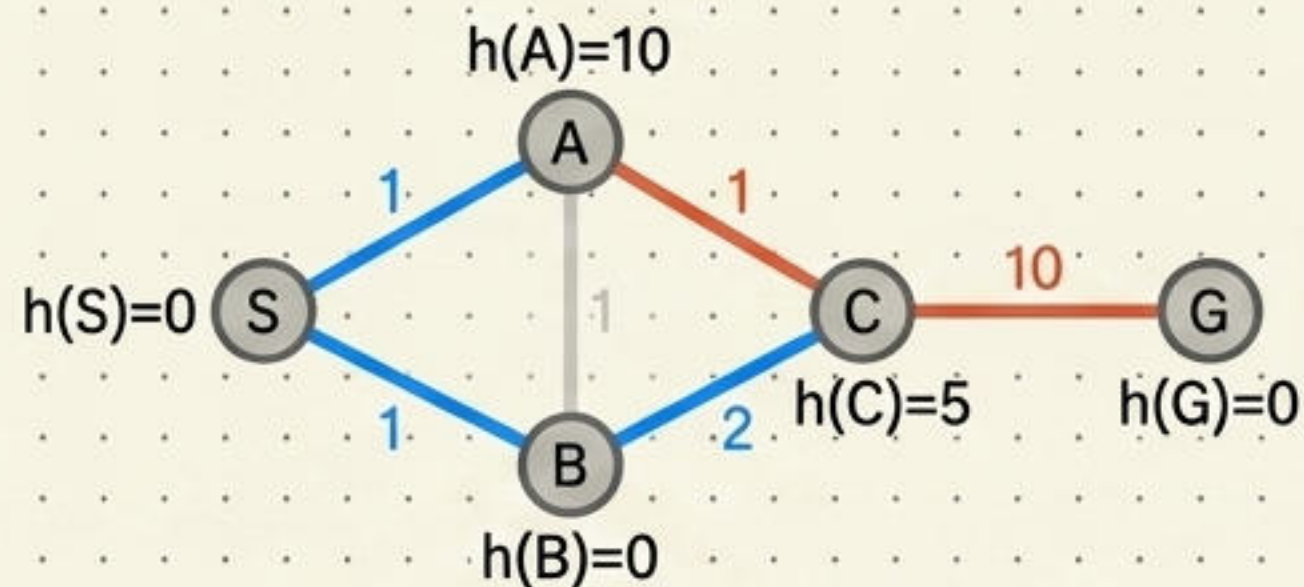
— מקיימת את אי-שוויון המשולש.

A* מול Greedy Search



הגרידי עלול ליפול למלכודות מקומיות כי הוא עיוור לעלות המסלול שעבר.
A*A מבטיח אופטימליות כי הוא משקלל את שני המדדים.

סכנות זיכרון: חיפוש עץ מול חיפוש גרף



מדוע עקביות קריטית?

בדיקת הקשת $A \rightarrow C$:

$$h(A) = 10, \text{ Cost} = 1, h(C) = 5$$

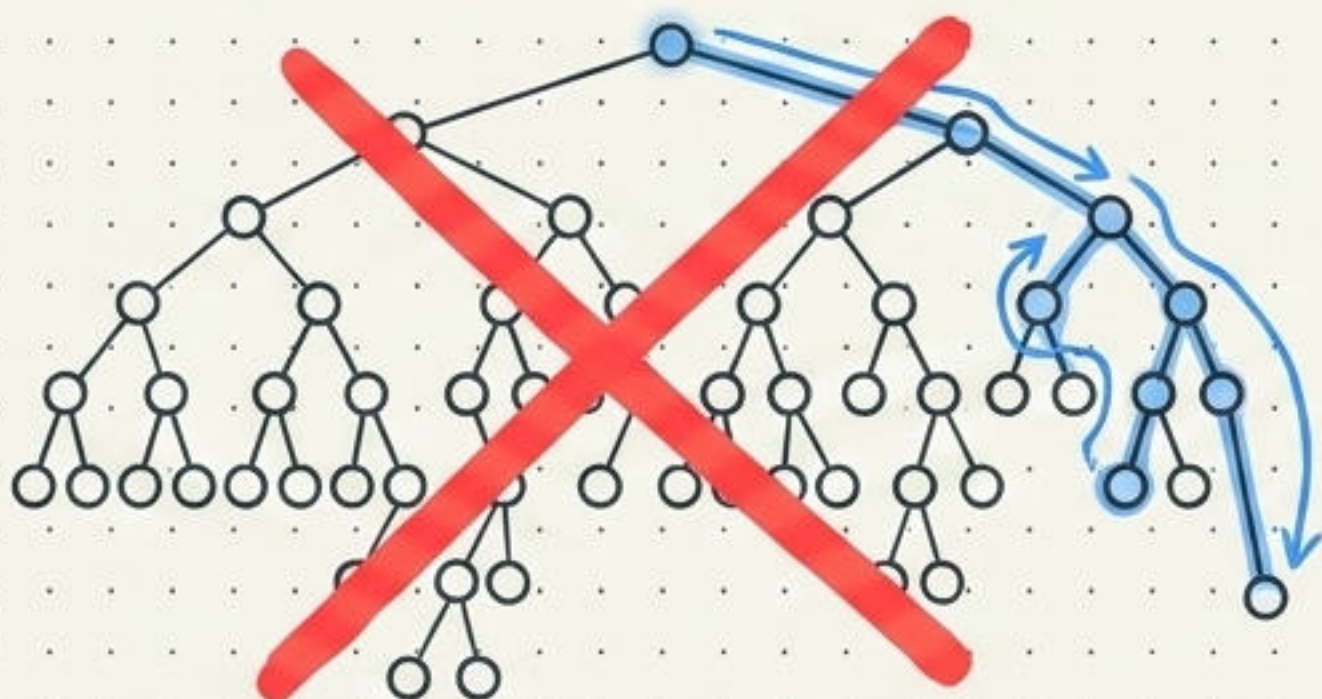
$$10 \leq 1 + 5 \quad \text{נוסחה:}$$

$$\text{תוצאה: } 10 \leq 6 \quad \text{❌ אי-עקביות!}$$

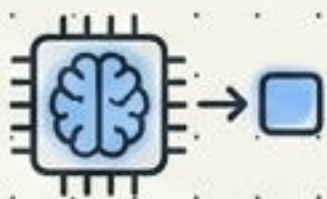
בחיפוש גרף (Graph Search), אלגוריתם A^* עלול לסגור את צומת C מוקדם מדי דרך מסלול תת-אופטימלי (דרך A), ולעולם לא יתקן את עצמו.

שינוי תפיסה: כשמטרה חשובה מהמסלול

חיפוש קלאסי



חיפוש מקומי



זיכרון קבוע

שומרים רק את המצב הנוכחי



אין חשיבות למסלול

מעניין אותנו רק הערך האופטימלי

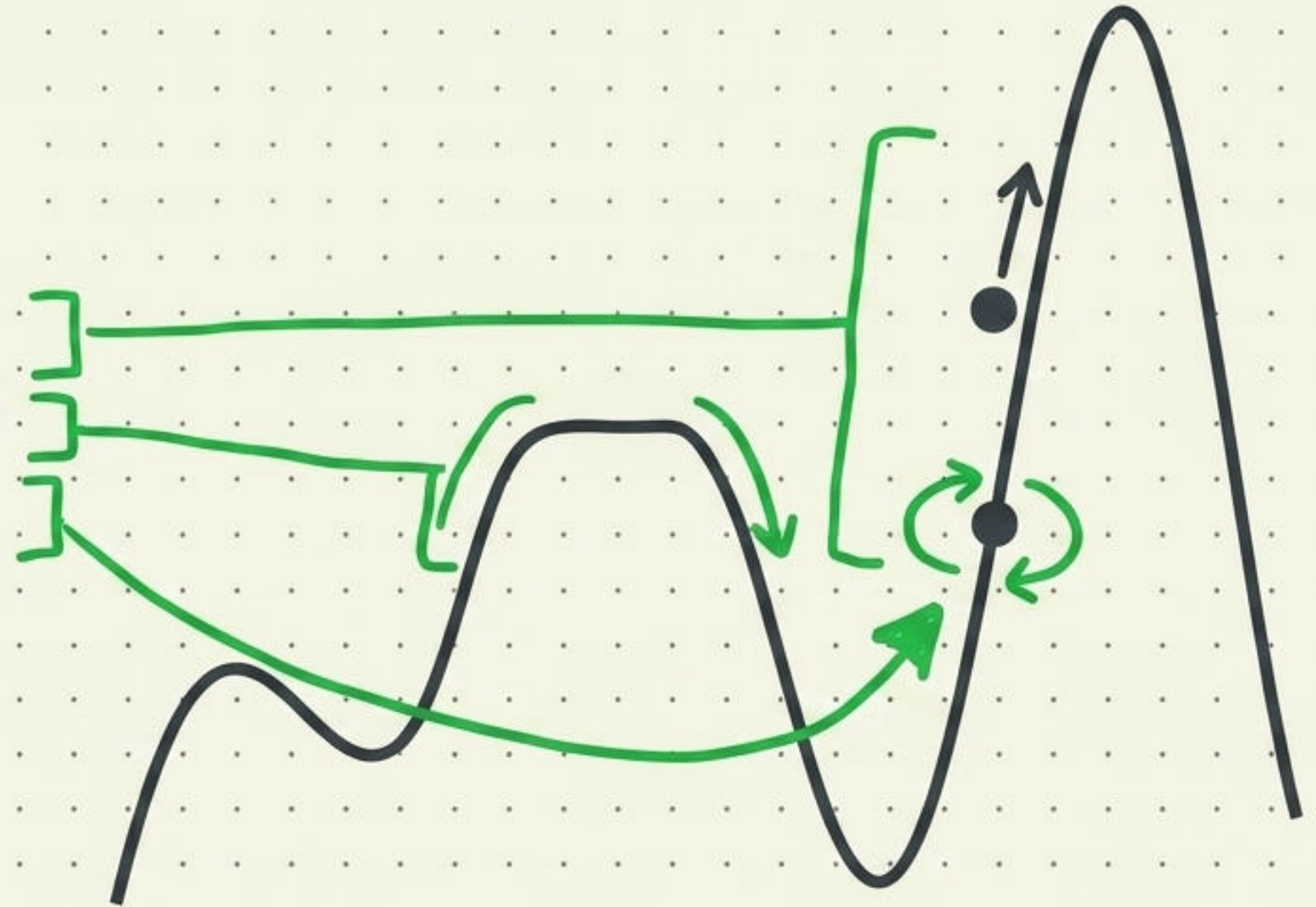


סביבות ענק

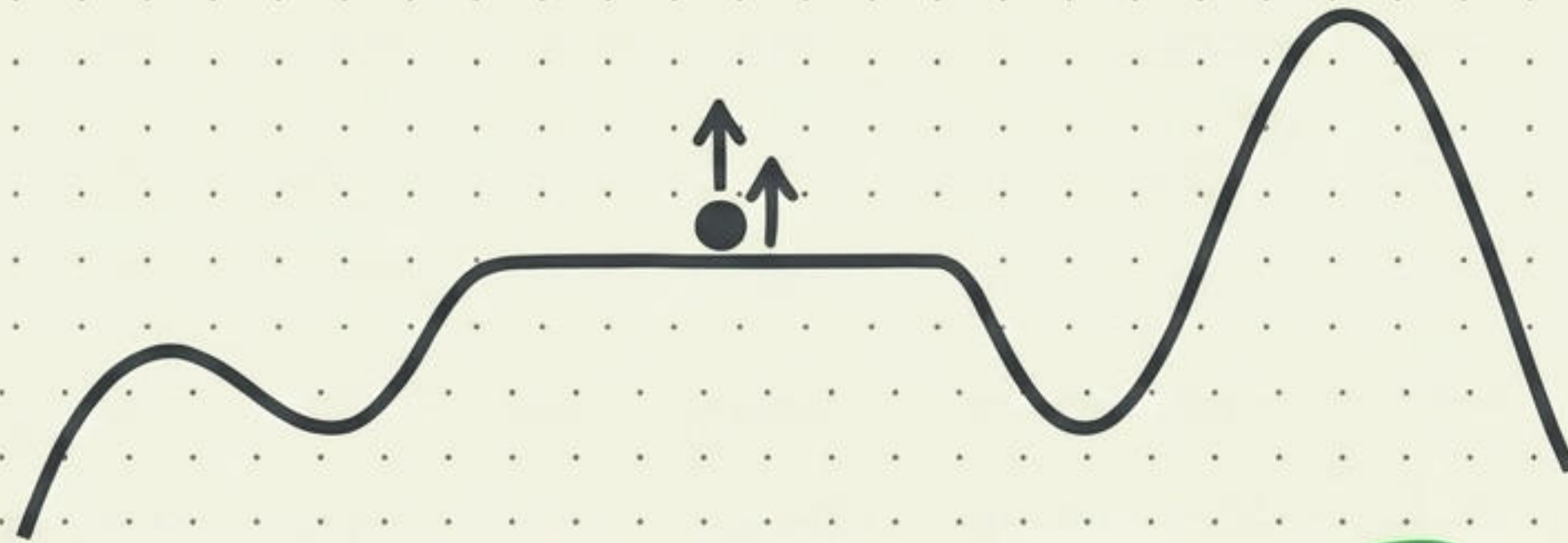
פתרון לבעיות שהפילו חיפוש קלאסי

טיפנוס גבעות: אופטימיזציה מקומית

```
current <- state  
loop for op in legal-operators  
  if better(new-val, best-val)
```



בעיית המישור והמהלכים הצדדיים



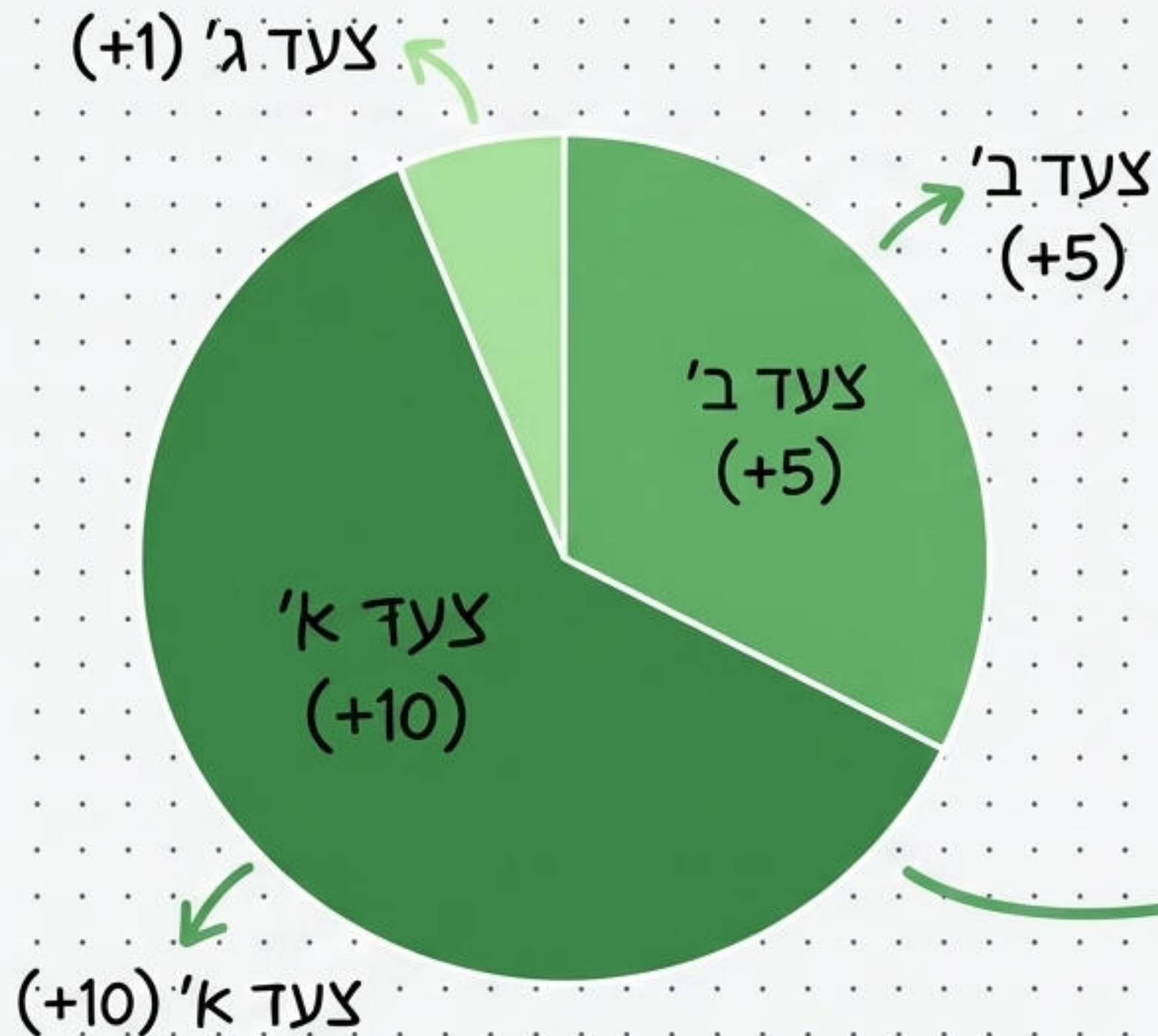
```
elseif best-val = h(current) AND sideways <= limit
```

הדילמה: מה עושים כשכל השכנים זהים?

הפתרון: מהלך צדדי (Sideways move).

הסכנה: לולאה אינסופית. חובה להגדיר limit לכמות הצעדים הצדדיים המותרים.

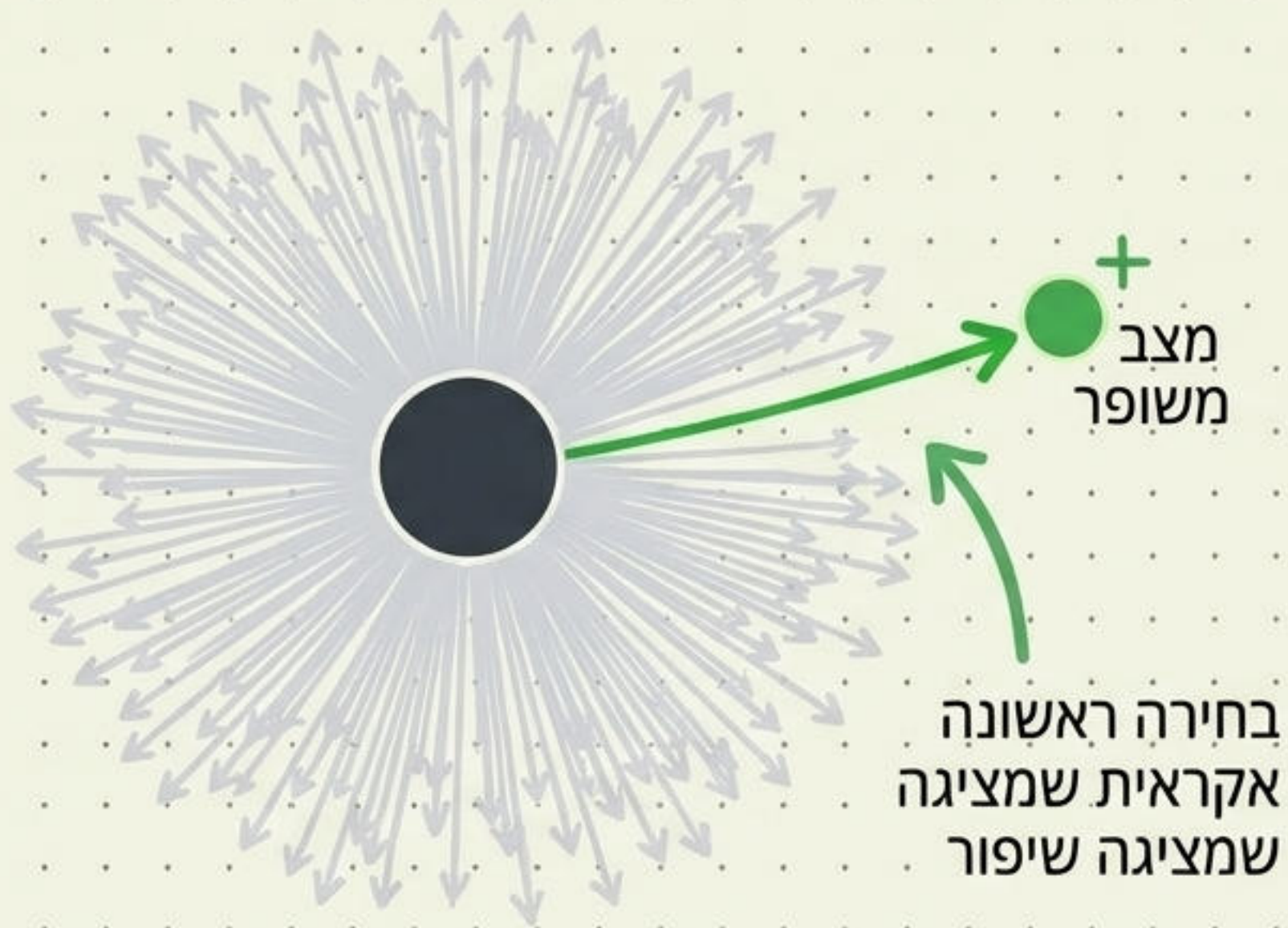
הכנסת אקראיות: טיפוס גבעות סטוכסטי



$$P_i = \frac{|\Delta v_i|}{\sum |\Delta v_i|}$$

לא תמיד בוחרים בצעד התלול ביותר!
הבחירה היא יחסית לעוצמת השיפור,
מה שמאפשר אלטרנטיבות ועקיפת
מכשולים.

התמודדות עם עומס: בחירה ראשונה



במקום לסרוק את כל השכנים...

בוחרים שכנים באקראי בזה אחר זה

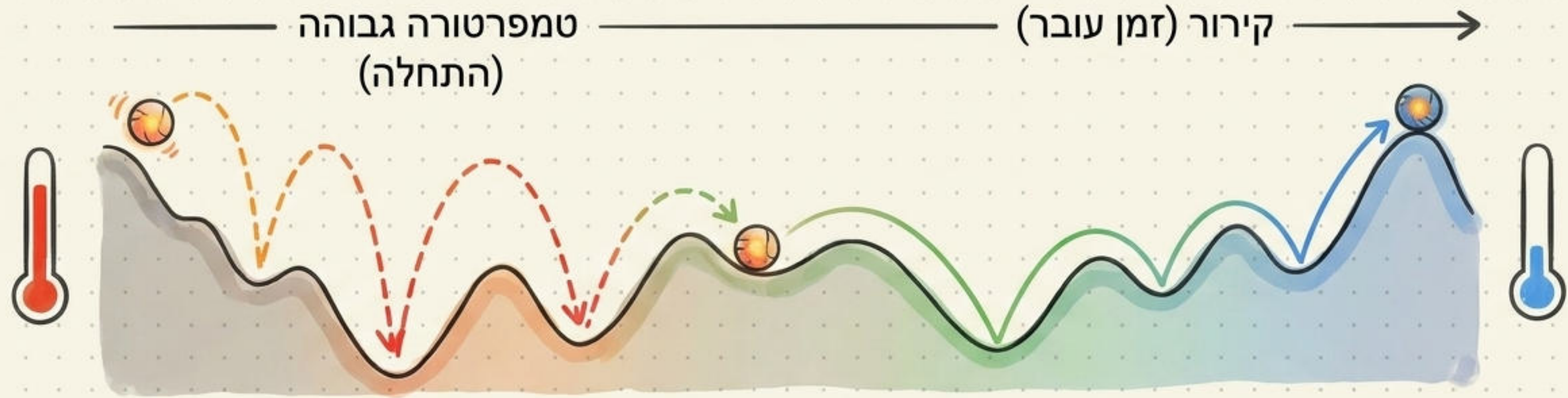
עוצרים ועוברים לשכן הראשון שמציג שיפור

```
if  $h(\text{new}) > h(\text{current})$  then  
  improved  $\leftarrow$  true
```

התנאי לשיפור

מושלם לבעיות עם מקדם סיעוף עצום, כמו לוח השחמט 8x8 מתחילת המצגת.

בריחה מהפסגה המקומית: Simulated Annealing



טמפרטורה (T) התחלתית:

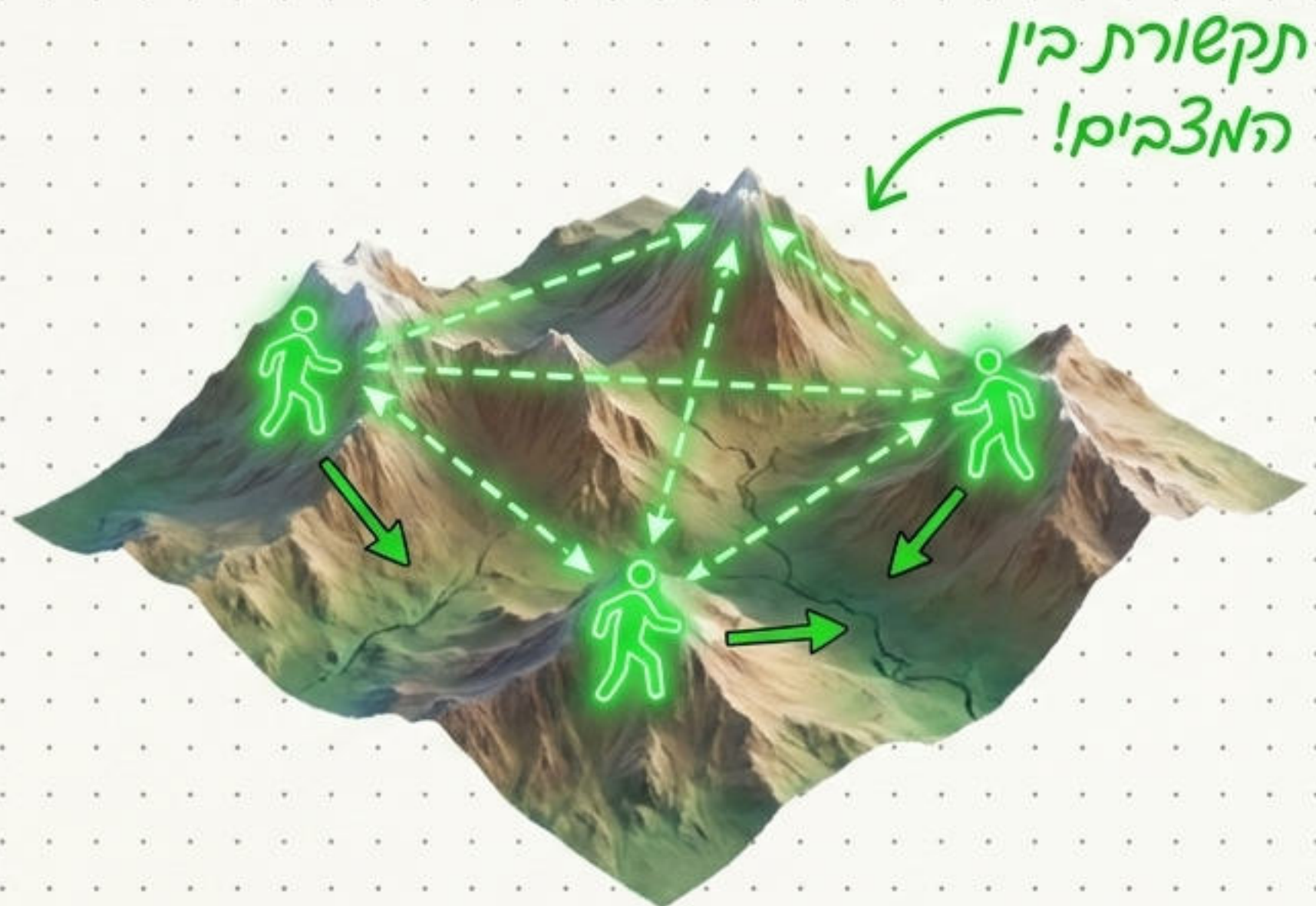
מתחילה גבוהה - מאפשרת הרבה הרבה צעדים גרועים כלפי מטה לחקר המרחב.

תהליך קירור (Cooling):

ככל שהזמן עובר, T יורד, והאלגוריתם הופך ל'חמדן' שרק עולה מעלה.

טיפ אסטודנט:
לעיתים צריך לרדת
כדי להגיע לפסגה
גבוהה יותר!

כוחה של קבוצה: חיפוש אלומה מקומי



- זה לא סתם k חיפושים במקביל!
- מצב טוב מגייס אליו את שאר החיפושים.
- אנלוגיה: ברירה טבעית (Natural Selection) – רק הצאצאים של המצבים המוצלחים ביותר שורדים לדור הבא.

קטלוג האלגוריתמים: טבלת סיכום לסטודנט

אלגוריתם	סוג החיפוש	זיכרון נדרש	האם אופטימלי?
A*	<u>מסלול</u>	גבוה	<u>כן</u> (אם עקבית/קבילה)
Greedy	<u>מסלול</u>	גבוה	<u>לא</u>
Hill Climbing	<u>מצב</u>	קבוע (מצב 1)	<u>לא</u> (מלכודת מקומית)
Stochastic HC	<u>מצב</u>	קבוע	<u>סיכוי משופר</u>
Beam Search	<u>מצב</u>	קבוע (k מצבים)	<u>הסתברות גבוהה</u>

הבנת המגבלות של כל אלגוריתם היא המפתח לבחירת הכלי הנכון לפתרון הבעיה.