Backtracking Search

- ואריאציה על DFS אשר משתמשת במדיניות LIFO ליצירת הצמתים במקום פיתוחם (במקום פיתוח כל הצמתים עבור כל צומת).
- כאשר נבחר צומת לפיתוח, מיוצר רק בן (successor) אחד ובן זה יהיה הצומת הבא לפיתוח.
- אם הצומת המפותח מקיים את אחד מתנאי העצירה, התוכנית תיסוג לאב הקדמון הקרוב ביותר שלא פותח (כלומר לאב הקדמון הקרוב ביותר בעל בנים שעדיין לא יוצרו).

Backtracking Search – cont'

- האלגוריתם:
  - שים את צומת ההתחלה ב- OPEN.
    (אם זהו צומת המטרה – סיים)
  - אם OPEN ריקה
    סיים – שגיאה.
    אחרת:
    המשך.
  - בדוק את הצומת העליון מ- OPEN (נקרא לצומת זה n).
  - אם העומק של n שווה לחסם העומק (depth bound) או שכל הענפים היוצאים מ-n כבר נחקרו:
    הסר את n מ- OPEN ועבור ל2-.
    אחרת:
    המשך.
  - צור בן חדש של n (מענף שעדיין לא נחקר) נקרא לבן זה n'.
    שים את n' בראש OPEN ושמור בו מצביע חזרה ל-n.
  - סמן ב-n כי הענף (n, n') נחקר.
  - אם n' הוא צומת המטרה:
    סיים. הפתרון הוא המסלול המתקבל הליכה
    לאחור מצומת המטרה דרך המצביעים לאבות.
    אחרת:
    המשך.
  - אם n' הוא מבוי סתום:
    הסר אותו מ- OPEN.
  - עבור ל- 2.

Backtracking Search – cont'

- OPEN היא רשימת הצמתים על מסלול השוטטות בעץ ומשמשת לצורך ביצוע הנסיגה (backtracking)

- יתרונות האסטרטגיה:
  - היתרון העיקרי של Backtracking על DFS הוא חסכון גדול יותר בזיכרון.
  - יתרון נוסף של Backtracking: אינו מפתח את הצמתים שנמצאים מימין למסלול הפתרון.

- חסרונות האסטרטגיה:
  - אסטרטגיה זו לא מסוגלת להיעזר באינפורמציה יוריסטית להערכת הבן המועמד לפיתוח (לעומת הגרסה המיודעת חלקית של DFS).

Breadth First Search (BFS)

- חיפוש לרוחב - ניתנת עדיפות לצמתים ה"רדודים" ביותר בגרף החיפוש. כלומר, נחקור ראשית צמתים הנמצאים בעומק זהה בגרף החיפוש.
- אסטרטגית BFS תמומש ע"י שימוש במדיניות FIFO. כלומר, נפתח ראשית את הצמתים ה"ישנים" יותר ברשימת ה- OPEN.
- לכן, בכל פעם שנפתח צומת נוסיף את בניו החדשים (new successors) בסוף רשימת OPEN.

Breadth First Search – cont’

- יתרונות האסטרטגיה:
  - מובטח כי בגרפים סופיים, האלגוריתם יסתיים וימצא פתרון אם קיים כזה.
  - מובטח כי ימצא הפתרון ה"רדוד" ביותר. (כלומר המסלול הקצר ביותר מצומת ההתחלה לצומת המטרה).

- חסרונות האסטרטגיה:
  - צריכת זיכרון גדולה מאוד: בכל זמן נתון זוכרים את כל הגרף שנחקר עד לנקודת זמן זו.

סה"כ מספר הצמתים שיפותחו:

$$\frac{B^{L-1}-1}{B-1}$$

  - חיפוש לא יעיל (מספר הצעדים אקספוננציאלי באורך הפתרון)

העמקה הדרגתית (Iterative Deepening)

- שילוב בין חיפוש לעומק (DFS) וחיפוש לרוחב (BFS).

- האלגוריתם:

  בצע חיפוש לעומק תחילה עם הגבלת עומק 1.

  אם הפתרון לא נמצא

  בצע חיפוש לעומק עם הגבלת עומק 2

  אם הפתרון לא נמצא

  בצע חיפוש לעומק עם הגבלת עומק 3

  :

  .

- האלגוריתם אינו זוכר דבר בין החיפושים (מלבד הגבלת העומק באיטרציה הקודמת).

ID – cont'

- יתרונות:
  - דרישת הזיכרון הינה לינארית באורך הפתרון.
  - הפתרון שנמצא הוא הקצר ביותר. (פתרון אופטימאלי)
  - האלגוריתם שלם.

- חסרונות:
  - בזבוז בזמן החיפוש – כל החיפושים של הרמות 1 עד L-1 מבוזבזים.

הערה: כיוון שרוב הצמתים של עץ מתרכזים בעלים, הבזבוז אינו גדול.

סה"כ מספר הצמתים שיפותחו:

$$L+(L-1)b+(L-2)b^2+\cdots+1b^L$$