אלגוריתם A*

אלגוריתם A* - הרעיון

- אלגוריתם חיפוש A* הוא אלגוריתם חיפוש מונחה היוריסטיקה על צמתי גרף, תוך חיפוש צומת המקיים תכונה מסוימת (צומת היעד).
- נעשה בו שימוש נרחב לבעיות הדורשות מציאת מסלולים בגרפים.
- מתחשבים ב- **עלות עד כה וגם בעלות משוערת עד למטרה.**
- אינטואיטיבית: הסכום של שני המחירים הוא המחיר הקובע.
- אם הפונקציה היוריסטית היא קבילה (admissible), שיטה זו נותנת פתרון אופטימאלי.

אלגוריתם A* - סימונים

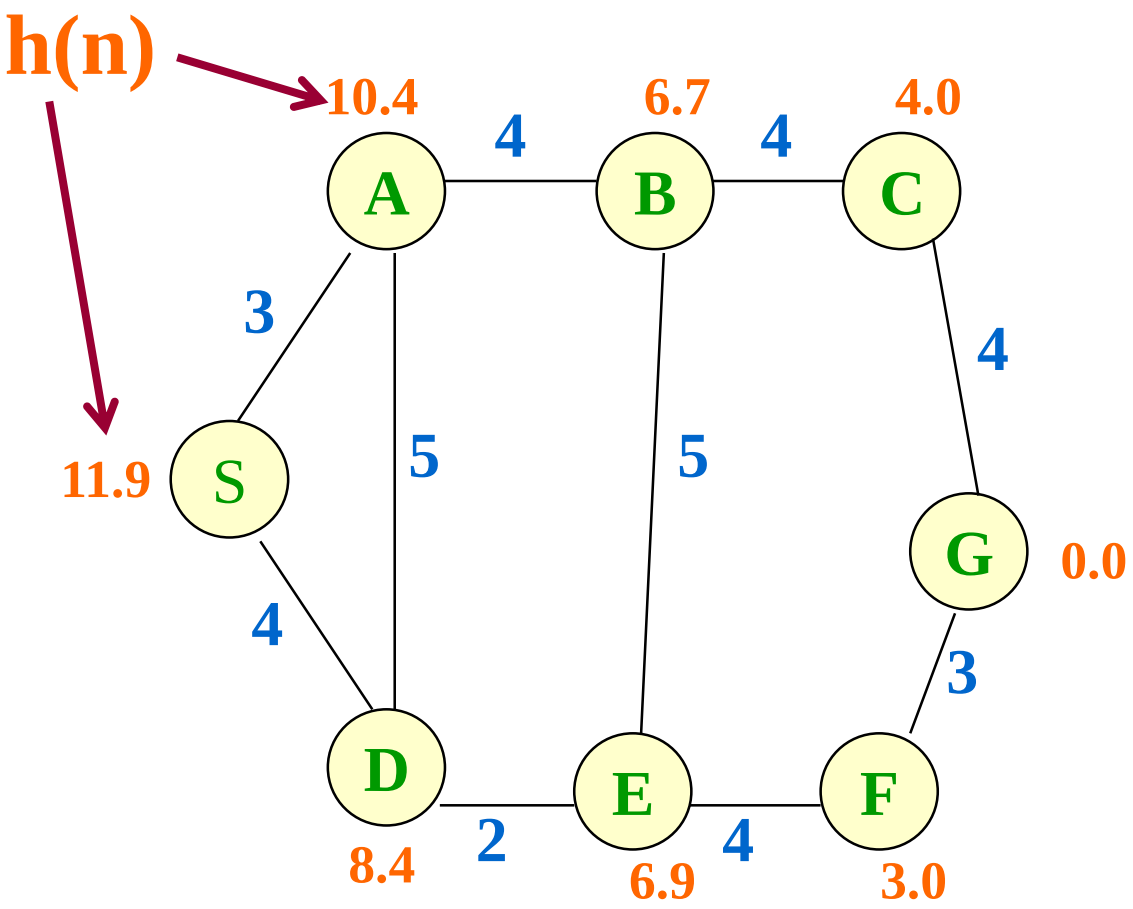
- שתי פונקציות
  - מחיר עלות מההתחלה : $g(n)$ - מדויק תמיד.
  - מחיר עלות עד למטרה : $h(n)$ - מחיר משוער
- פונקציה היוריסטית : $f(n)=g(n)+h(n)$
- האסטרטגיה : $\min(f(n))$
- מחיר המסלול האופטימלי : f^*
- מחיר המסלול האופטימלי העובר דרך צומת n:

$f^*(n),h^*(n)$

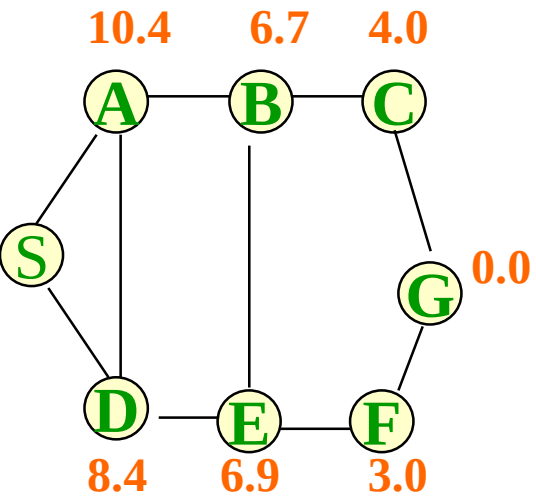
אלגוריתם A* - סימונים (המשך)

- <u>דרישה</u>: המחיר המשוער $h(n)$ תמיד **קטן** מהמחיר האמיתי, כלומר: $h(n) <= h^*(n)$
- במקרה זה נאמר ש h -היא הערכה מלמטה.
- תמיד מתקיים: $g(n) = g^*(n)$
- לכן: (נובע משתי הנקודות האחרונות) $f(n) <= f^*(n)$

מרחק מהצומת אל המטרה בקו ישר



דוגמא לחיפוש בגרף

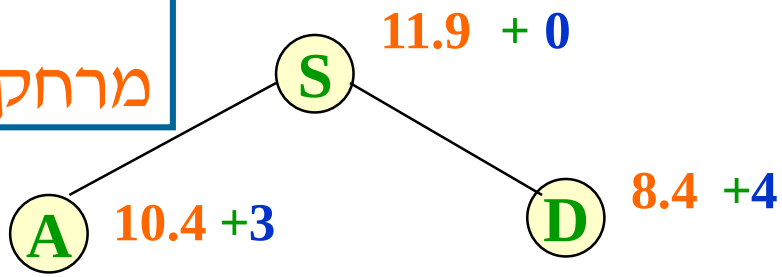


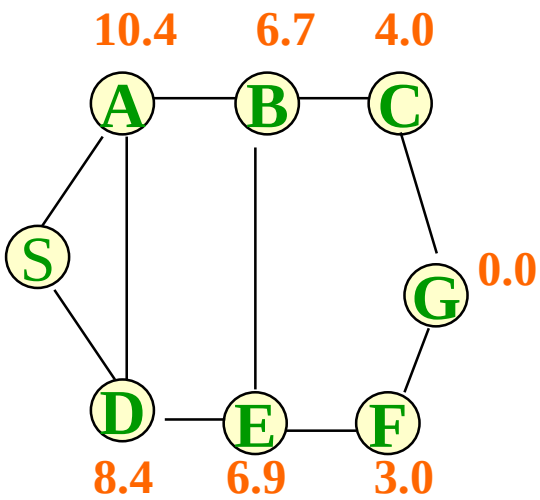
דוגמא לחיפוש בגרף

Heuristic function:

המרחק עד כה +

מרחק בקו ישר מהמטרה



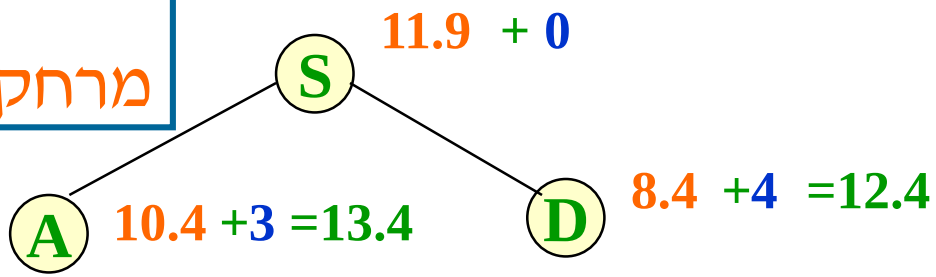


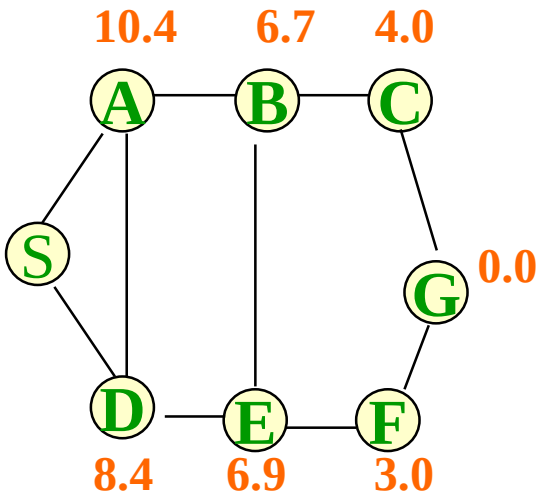
דוגמא לחיפוש בגרף

Heuristic function:

המרחק עד כה +

מרחק בקו ישר מהמטרה



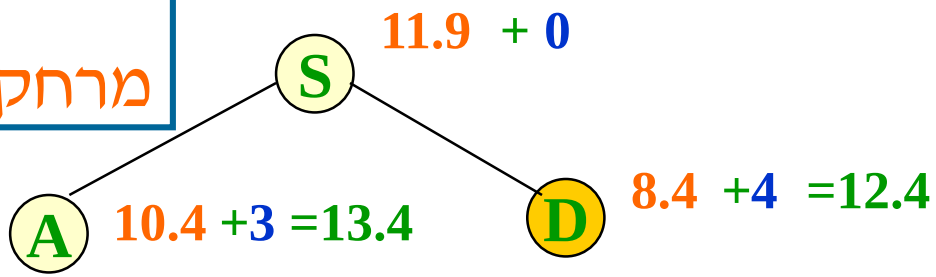


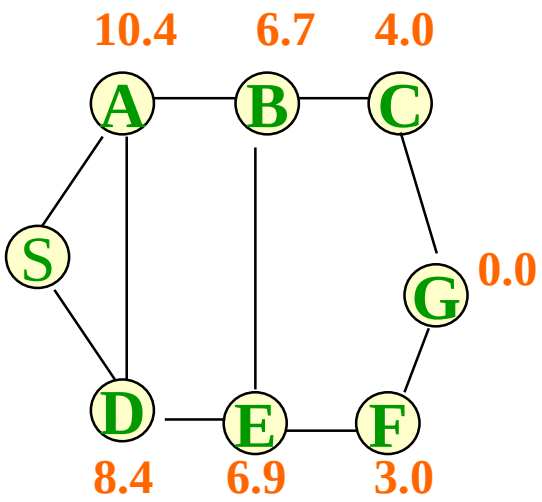
דוגמא לחיפוש בגרף

Heuristic function:

המרחק עד כה +

מרחק בקו ישר מהמטרה



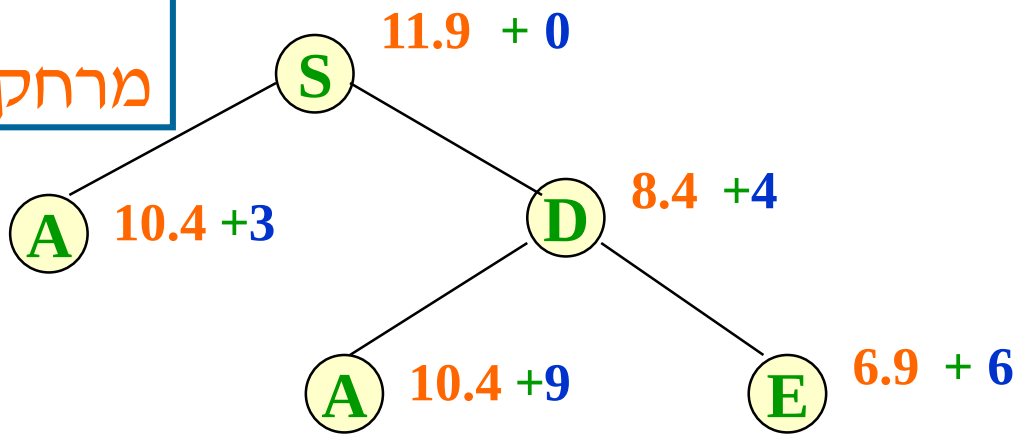


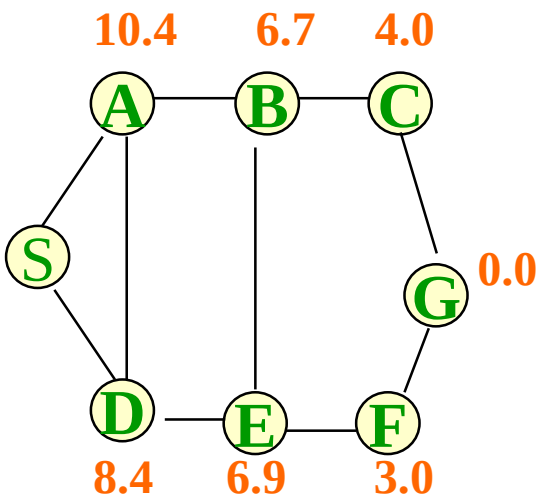
דוגמא לחיפוש בגרף

Heuristic function:

המרחק עד כה +

מרחק בקו ישר מהמטרה



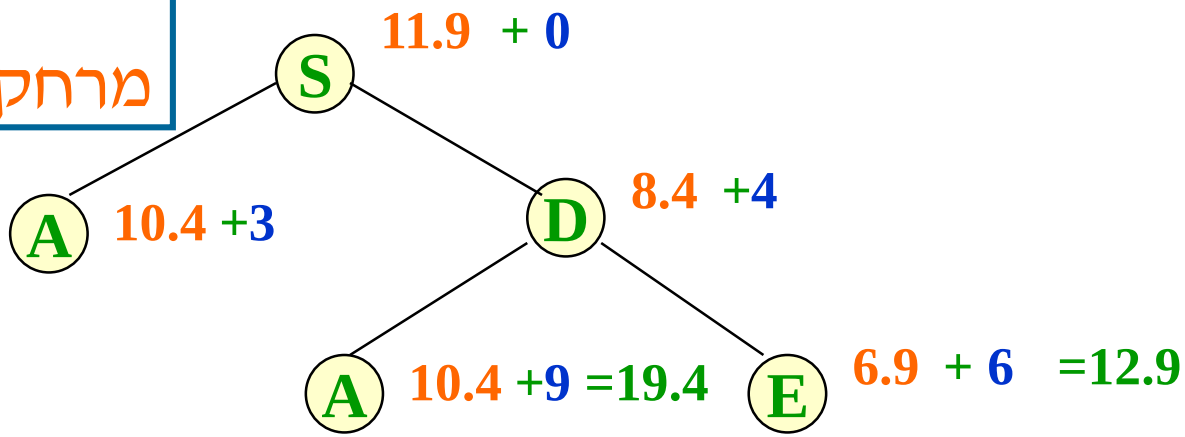


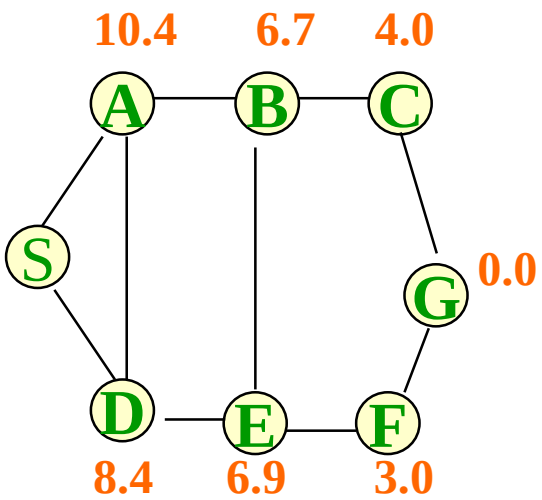
דוגמא לחיפוש בגרף

Heuristic function:

המרחק עד כה +

מרחק בקו ישר מהמטרה



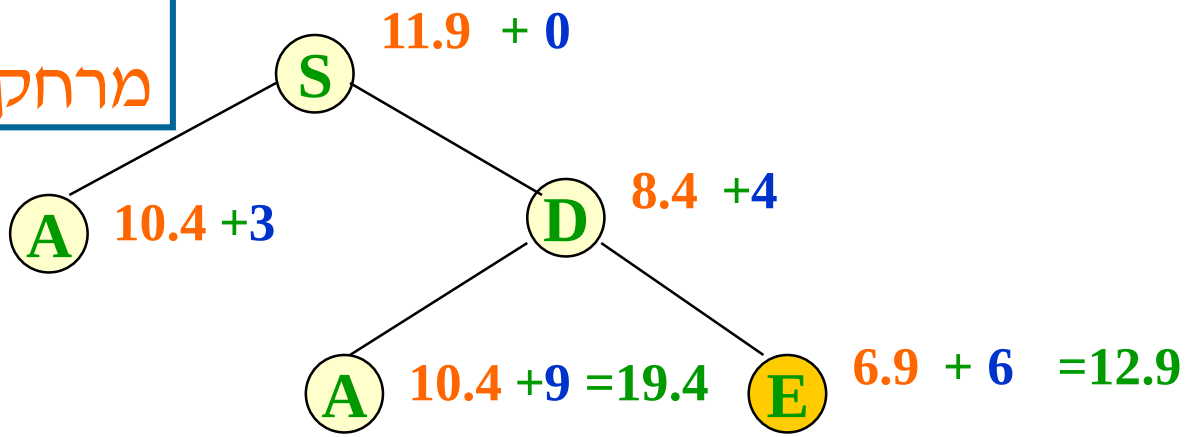


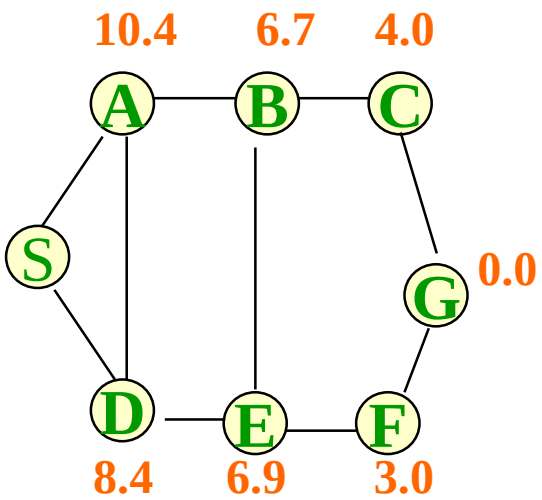
דוגמא לחיפוש בגרף

Heuristic function:

המרחק עד כה +

מרחק בקו ישר מהמטרה



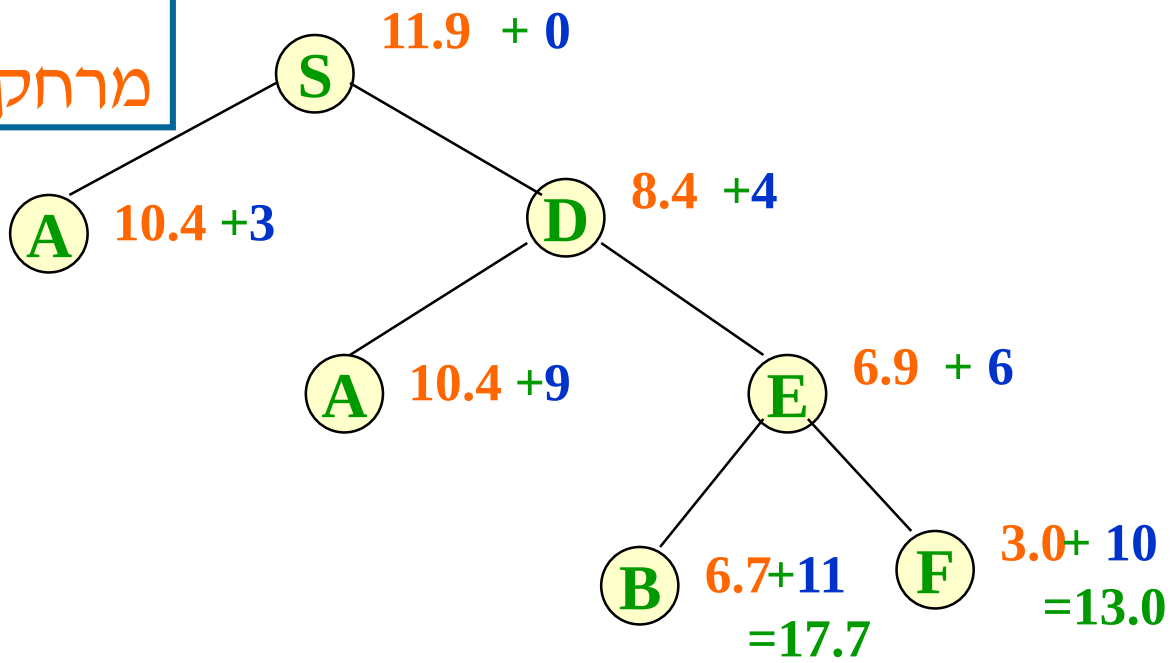


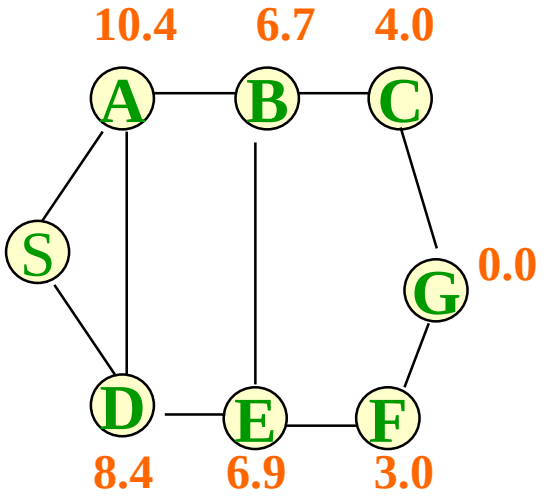
דוגמא לחיפוש בגרף

Heuristic function:

המרחק עד כה +

מרחק בקו ישר מהמטרה



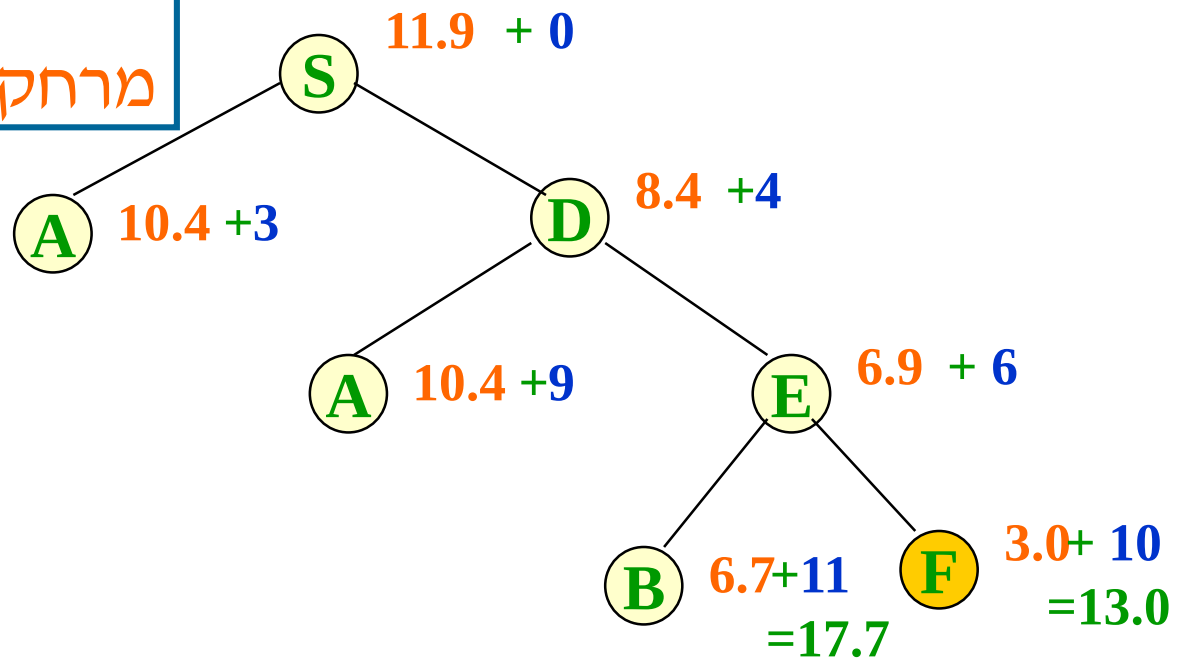


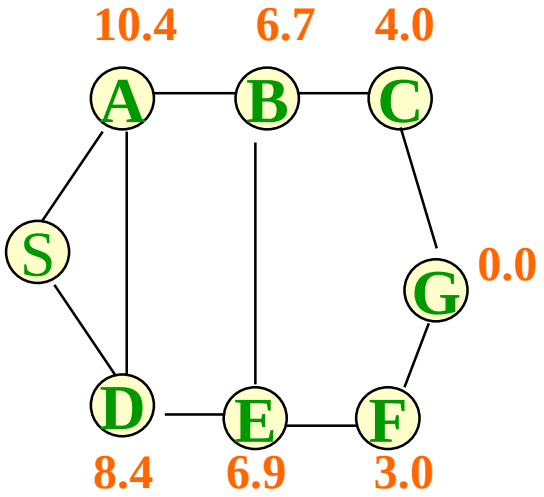
דוגמא לחיפוש בגרף

Heuristic function:

המרחק עד כה +

מרחק בקו ישר מהמטרה



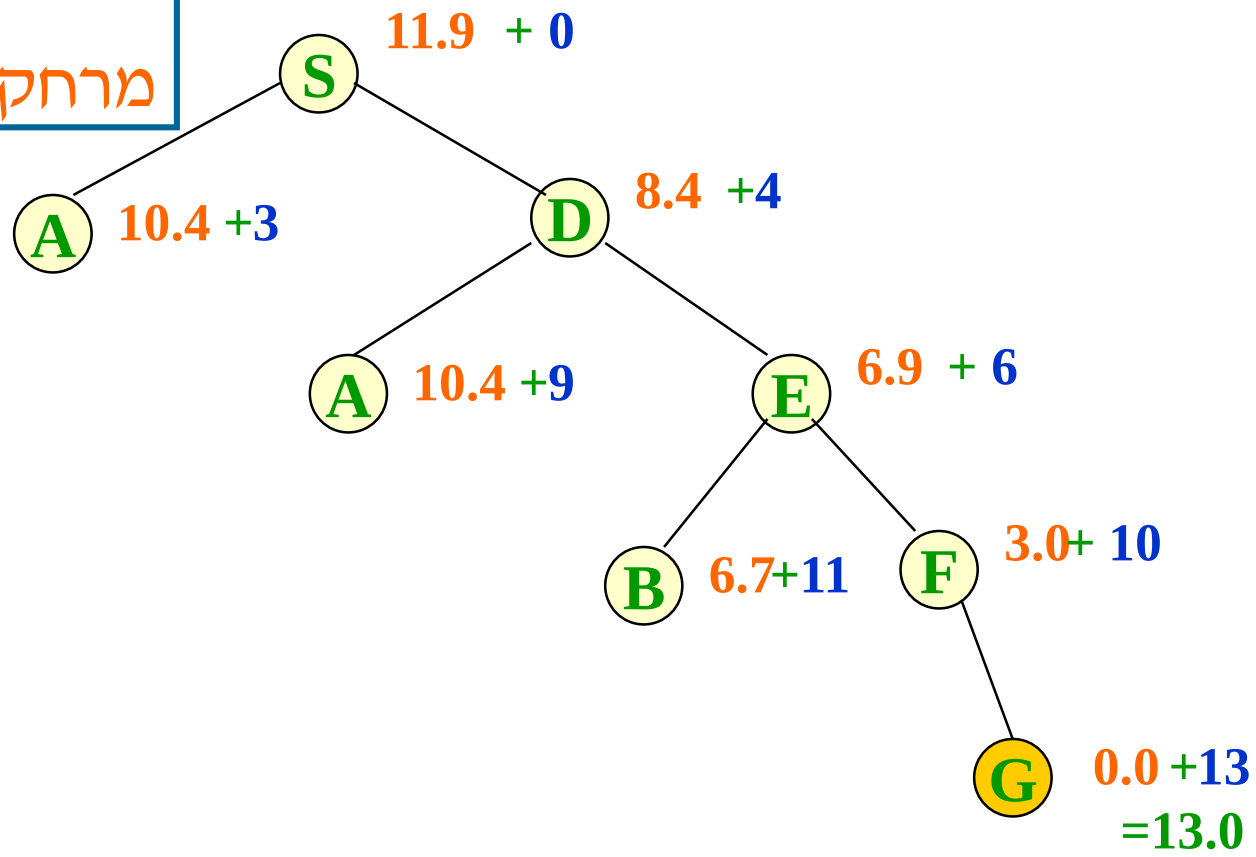


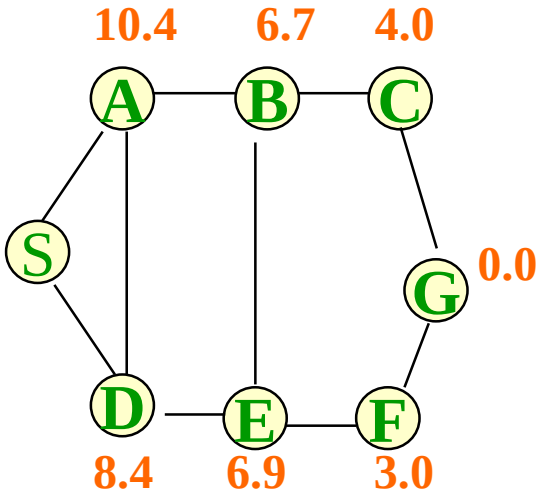
דוגמא לחיפוש בגרף

Heuristic function:

המרחק עד כה +

מרחק בקו ישר מהמטרה





השוואת פונקציה היוריסטית

- הערכה לא מוצלחת של "המרחק" שנשאר למטרה תגרום לעבודה מיותרת.
- בהינתן שני אלגוריתמים A_1 ו A_2 -עם פונקציות היוריסטיות $h_1(n), h_2(n) <= h^*(n)$

איזו מהם טובה יותר?

השוואת פונקציה היוריסטית

- אם $h_1(n) <= h_2(n)$ עבור כל צומת n שאינה צומת מטרה, אז נאמר ש h_2 -שולטת על h_1 .
- <u>טענה</u>: אם h_2 שולטת על h_1, אזי מספר הצמתים שיורחבו ע"י A^* המשתמש ב h_1 -לפחות כמספר הצמתים שיורחבו ע"י A^* המשתמש ב-h_2.

פונקציות היוריסטיות - דוגמא פזל 8

5	4	
6	1	8
7	3	2

מצב התחלתי-s

1	2	3
8		4
7	6	5

מצב המטרה

- h_1: מספר המשבצות שאינן במקום הנכון.
- h_2: סכום "מרחקי מנהטן" ממקום המשבצת הנוכחי אל המקום במצב המטרה.

$h_1(s) = 7$

$h_2(s) = 2+3+3+2+4+2+0+2=18$

פונקציות היוריסטיות - דוגמא פזל 8

- איזו פונקציה טובה יותר?

השוואת ביצועי h_1 וh_2 -

	Search Cost			Effective Branching Factor		
d	IDS	A*(h_1)	A*(h_2)	IDS	A*(h_1)	A*(h_2)
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	364404	227	73	2.78	1.42	1.24
14	3473941	539	113	2.83	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

Figure 4.8 Comparison of the search costs and effective branching factors for the ITERATIVE-DEEPENING-SEARCH and A* algorithms with h_1, h_2. Data are averaged over 100 instances of the 8-puzzle, for various solution lengths.

כיצד להמציא פונקציות היוריסטית

- לחשוב על גרסה מקלה של הבעיה
  - להוריד מגבלות
  - פזל-8 : הזזת משבצת גם למשבצת תפוסה.
  - חיפוש בגרף : מרחק בקו ישר.
- לחבר מספר פונקציות:

$$f(n) = F(f_1(n), f_2(n), \ldots, f_k(n)) \ , \ F = \max, \text{sum}$$

- חשוב לדאוג לעלות חישוב נמוכה של הפונקציה היוריסטית

A-Star

```
function A*(start, goal) returns solution sequence

closedset := the empty set // The set of nodes already evaluated.
openset := {start} // The set of tentative nodes to be evaluated
came_from := the empty map // The map of navigated nodes.

g_score[start] := 0 // Cost from start along best known path.
// Estimated total cost from start to goal through y.
f_score[start] := g_score[start] + heuristic_cost_estimate(start, goal)
```

(continued on next slide)

A-Star

```
while openset is not empty
    current := the node in openset having the lowest f_score[] value
    if current = goal
        return reconstruct_path(came_from, goal)
    remove current from openset
    add current to closedset
    for each neighbor in neighbor_nodes(current)
        if neighbor in closedset
            continue
        tentative_g_score := g_score[current] + dist_between(current,neighbor)

        if neighbor not in openset or tentative_g_score < g_score[neighbor]
            came_from[neighbor] := current
            g_score[neighbor] := tentative_g_score
            f_score[neighbor] := g_score[neighbor] +
heuristic_cost_estimate(neighbor, goal)
            if neighbor not in openset
                add neighbor to openset
return failure
```

A-Star

```
function reconstruct_path(came_from,current)
    total_path := [current]
    while current not null
        current := came_from[current]
        total_path.append(current)
    return total_path
```

A-Star Demo

- מציאת מסלול קצר ביותר עבור רובוט מנקודת התחלה (האדומה) לנקודת סיום (הירוקה) באמצעות A-Star