הסתעף וחסום- Branch and Bound

בעיה קשה

בעיה קשה היא בעיה שאין עבורה אלגוריתם יעיל אשר מחשב יכול לפתור בזמן סביר (פולינומיאלי).

איך מתמודדים עם בעיה קשה?

- פתרון אופטימלי לבעיות עם **קלט קטן**: באמצעות תכנות לינארי או באמצעות הסתעף וחסום.
- עבור בעיות עם **קלט גדול**: באמצעות אלגוריתם היוריסטי- מציאת תשובה טובה או מהירה אך לא בהכרח אופטימלית, למשל- נשתמש באלגוריתם הדסון מור למזעור סכום הפיגורים $\sum T_j$ מכיוון שידוע שהוא ממזער את מספר הפיגורים $\sum U_j$.
  או באמצעות אלגוריתמים מטה- היוריסטיים- אלגוריתמים איטרטיביים סטוכסטיים אשר ככל שירוצו זמן רב יותר יתכנסו לכיוון הפתרון האופטימלי.

הסתעף וחסום-

שיטה המבוססת על פיתוח עץ המכיל את כל אפשרויות הזימון (n!). פיתוח העץ נעשה בצורה דינאמית. בכל פעם מפתחים חלקים מסוימים (הסתעף) וקוטמים חלקים אחרים (חסימה). מטרת הקטימה היא לחסוך בחינה של זימונים.

שלבי העבודה:

1. **מציאת חסם עליון- upper bound.** חסם עליון הוא ערך שהפתרון האופטימלי בהכרח לא גבוה ממנו. כל פתרון אפשרי מהווה חסם עליון אבל נרצה למצוא חסם עליון הדוק(קטן) ככל האפשר על מנת שנוכל לקטום ענפים בקלות בעזרתו ולהקטין את מרחב החיפוש באופן משמעותי.
2. **מגדירים משתנה עזר q_j.** זמן הסיום של הג'וב(q) במקום הj.   (Cj)
   בעבודה מהסוף להתחלה- $q_{[j]}$- סכום הזמנים של הג'ובים שטרם שובצו.
   min(max(Erj))
   33 ← $q_{[j]} = \sum t_j$
   $q_{[j]} = q_{[j+1]} - t_{[j+1]}$
   בעבודה מההתחלה לסוף- $q_{[j]}$- סכום הזמנים של הג'ובים ששובצו.
   $q_{[0]} = 0$
   $q_{[j]} = q_{[j-1]} + t_{[j]}$
3. **חסם תחתון בשלב ה [j] ($LB_{[j]}$)** : חסם תחתון הוא ה"קנס" שנצבר על ידי הג'ובים שכבר שיבצנו בפתרון הנוכחי. אם נבחר בפתרון שמתבסס על הכיוון הנוכחי ערך המדד בהכרח יהיה גדול או שווה לערך זה.
   נוסחת המעבר- בבעיות מסוג מינימום של סכום- $LB_{[j]} + LB_{[j-1]}$. $\boxed{LB_j=LB_{\text{קודם}}+\text{קנס נוכחי}}$
   בבעיה מסוג מינימום של מקסימום- ("הקנס" הנוכחי, $LB_{[j]} = \max(+LB_{[j-1]}$. ($[j-1]$- הכוונה הקודם).
   $\boxed{LB_j=\max\{LB_{\text{קודם}}, \text{קנס נוכחי}\}}$ ←
4. **שיטת הפרישה-** מההתחלה לסוף או מהסוף להתחלה.
   נבחר בכיוון הפרישה מתוך מטרה לפסול ענפים רבים ככל האפשר מוקדם ככל האפשר. לכן, אם המדד צפוי לקבל ערכים גבוהים בתחילת הסידור (למשל בהקדמות) אז נפרוש מההתחלה לסוף.
   ואם המדד צפוי לקבל ערכים גבוהים בסוף הסידור (למשל בפיגורים) נפרוש מהסוף להתחלה.
5. **קטימת הענפים-** שני תנאים:
   - $LB_i \geq UB$
   - כלל הדומיננטיות- אם קיימת קבוצה B שעבורה ערך המדד (LB_i) הוא x וקיימת קבוצה A המוכלת בB שעבורה ערך המדד גדול או שווה לx ברור שאין טעם לבדוק את הכיוון של A כיוון שבהכרח נקבל פתרון טוב יותר על ידי קבוצה B.
6. **שיטות חיפוש-** לפי מה נחליט לאיזה ענף להמשיך לבדוק. יש שתי שיטות-
   - *Jump Tracking*- נפרוש לפי הLB הנמוך ביותר מכל העץ.
   - *Back Tracking* - נפרוש לפי הLB הנמוך ביותר באותו השלב.

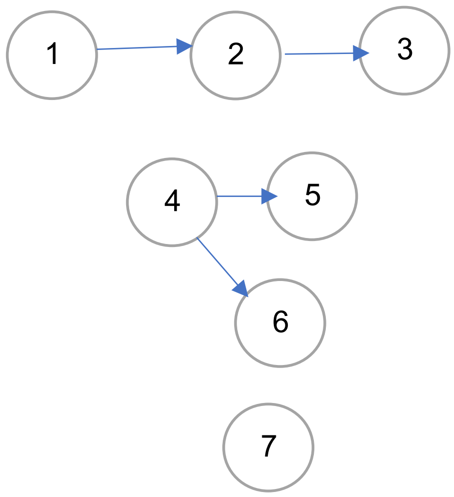
$LB = \max\{0, C_j - d_j)$
↓
q_j

דוגמה-

j	1	2	3	4	5	6	7
t_j	2	7	3	8	5	4	8
d_j	10	12	18	15	20	26	28
$t_j + d_j$	2+10=12	19	21	23	25	30	36

יש למזער את $\sum T_j$ על ידי הסתעף וחסום. עבוד בשיטת *Jump Tracking*.

מצא *UP* על פי הסידור המתקבל בסדר עולה של $t_{[j]} + d_{[j]}$.



פתרון-

איזה ג'וב יכול להיות משובץ?(לפי הג'וב שהז שלו הכי נמוך)	C_j	$T_j = \max(0, C_j - d_j)$
1,4,7	2	Max (0,2-10) =0
2,4,7	2+7=9	Max (0,9-12) =0
3,4,7	9+3=12	0
4,7	20	Max (0,20-15) =5
5,6,7	25	5
6,7	29	3
7	37	9

$$\sum T_j = 22 = UB$$

אנחנו רוצים למזער פיגורים, לכן נעבוד מהסוף להתחלה-

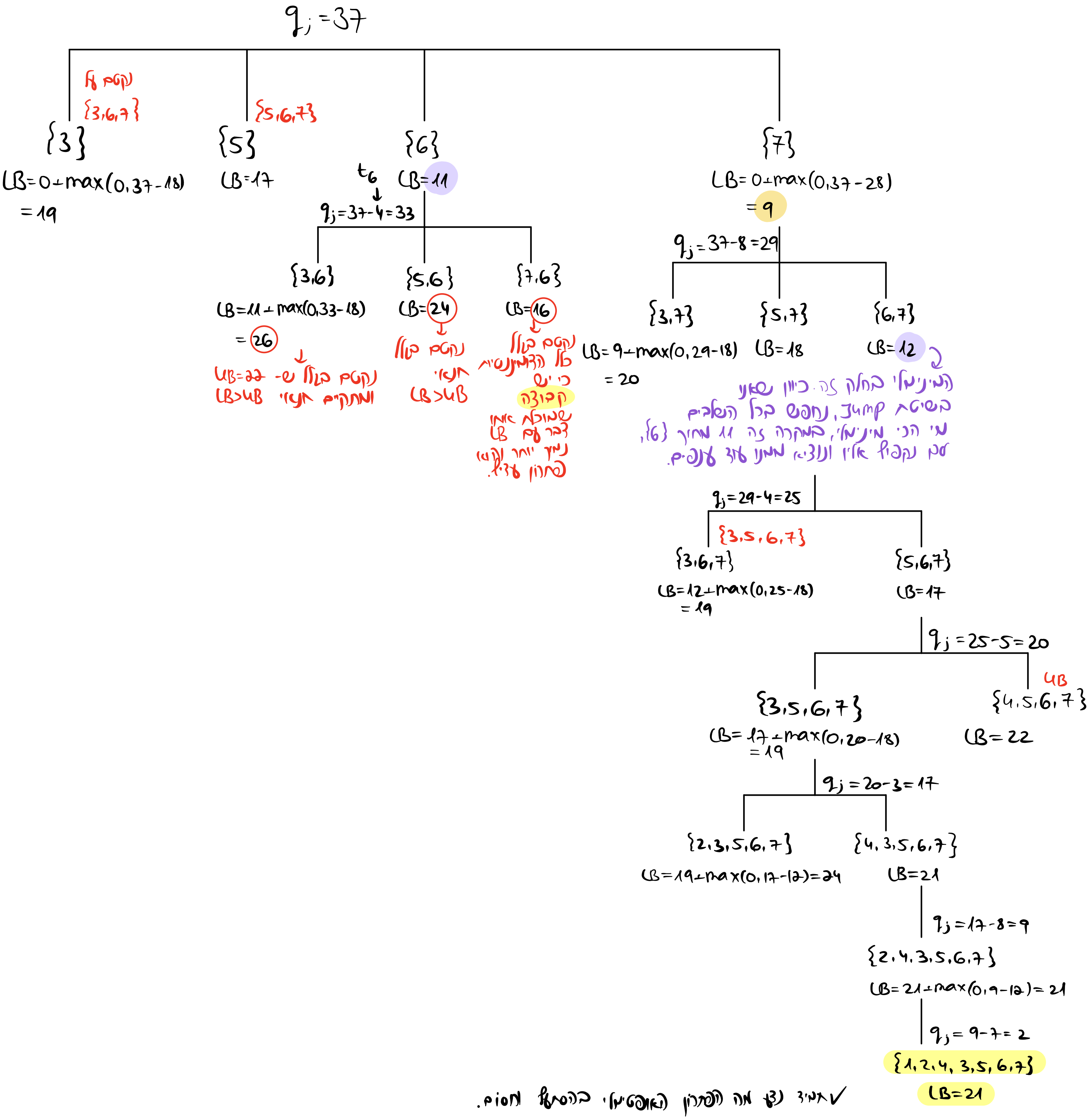
$$q_{[7]} = \sum t_j = 37$$

$$LB = LB_{\text{הקודם}} + \text{הקנס הנוכחי}$$

$$= LB_{\text{הקודם}} + T_j$$

$$= LB_{\text{הקודם}} + \max(0, C_j - d_j)$$

$$= LB_{\text{הקודם}} + \max(0, q_j - d_j)$$



✓תמיד נדע מה הפתרון האופטימלי בהסתעף וחסום.