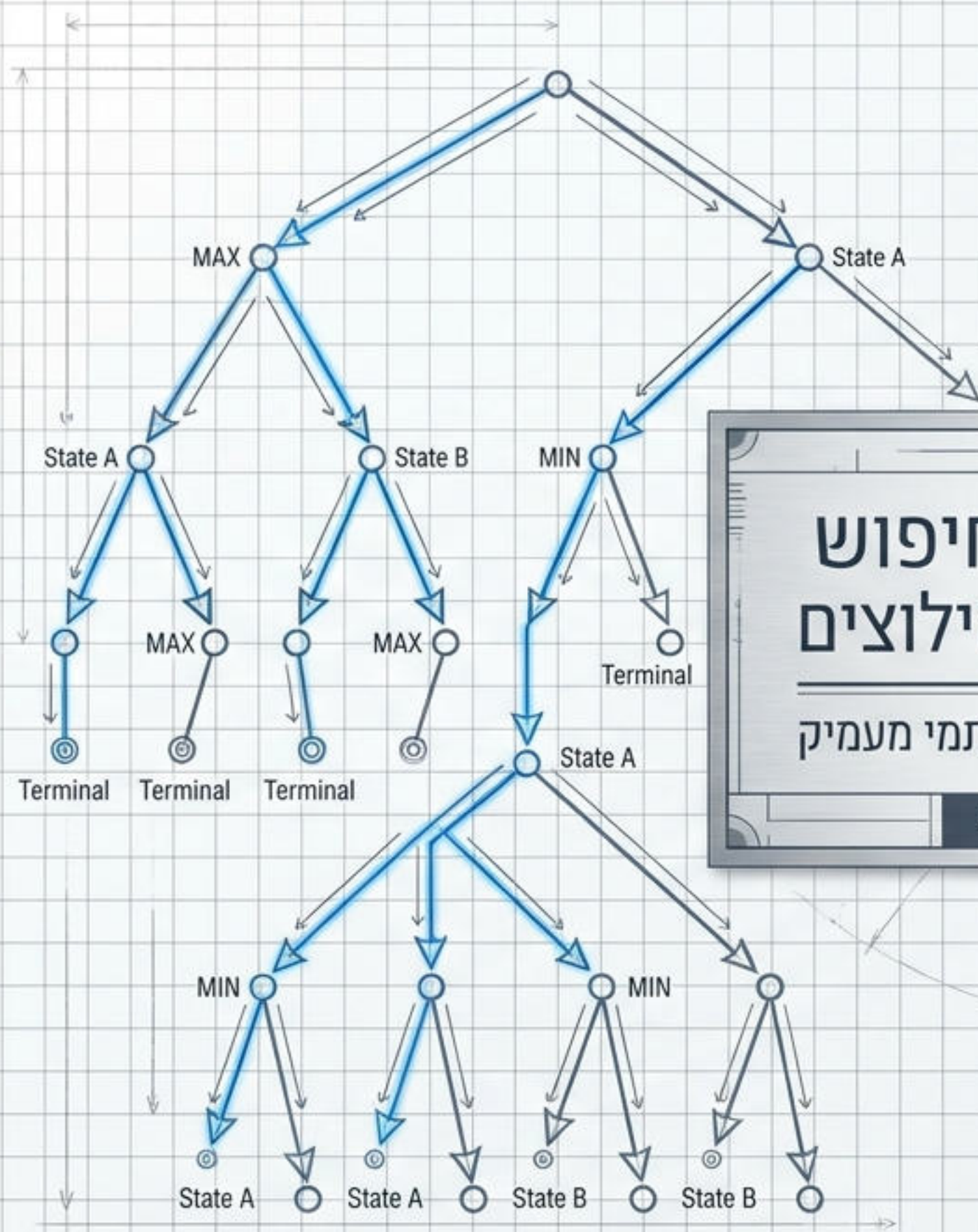




SCALE: 1:100

Variable X

Variable Y

Constraint C1

Constain D1

Variable Y

Constraint C2

Domain D1

Variable Y

Variable X

Constraint C2

Constraint C1

Solution

Variable Y

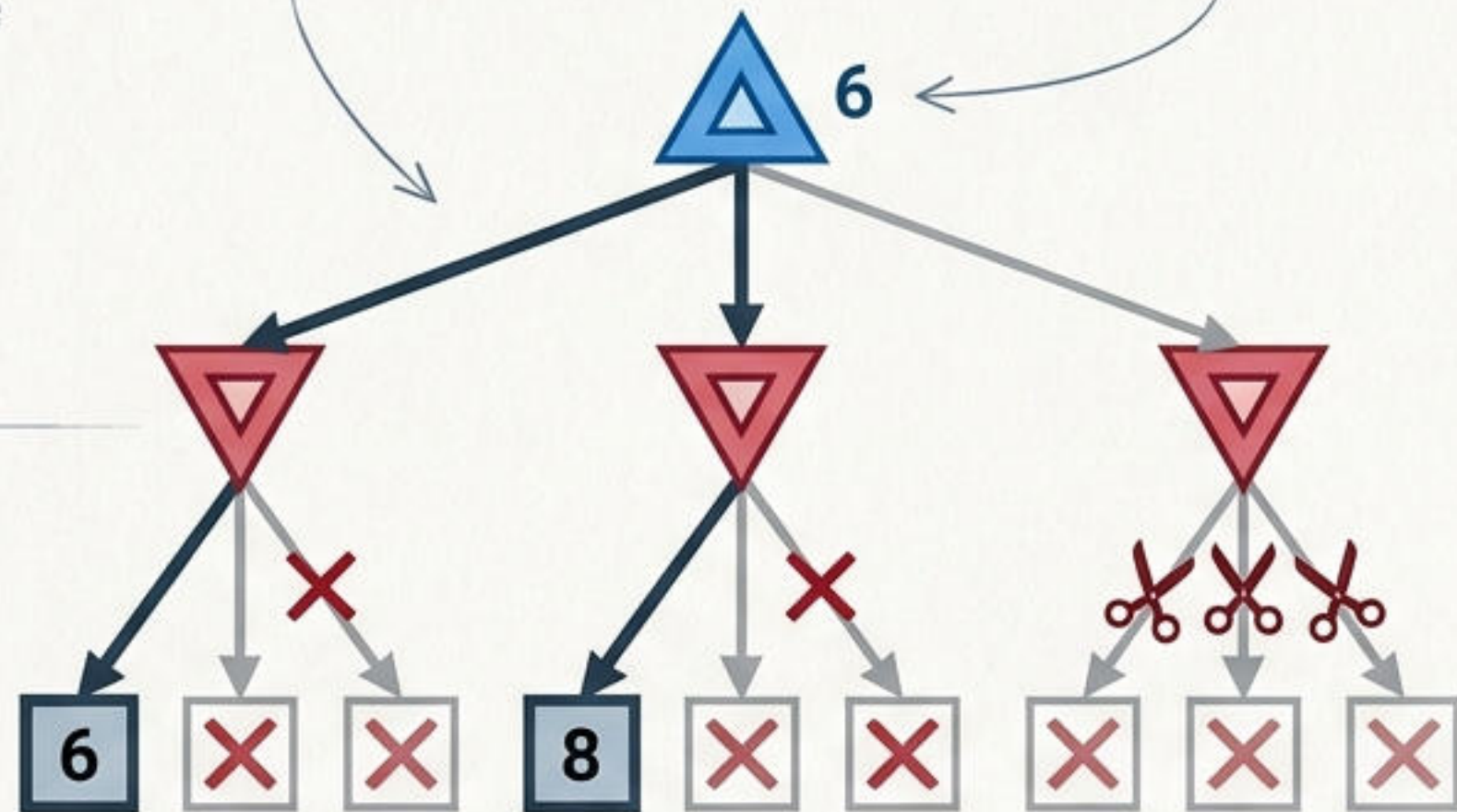
SECTION A-A

PLAN VIEW

חוק הברזל של גיזום אלפא-ביתא

Alpha-Beta Pruning

גיזום אלפא-ביתא

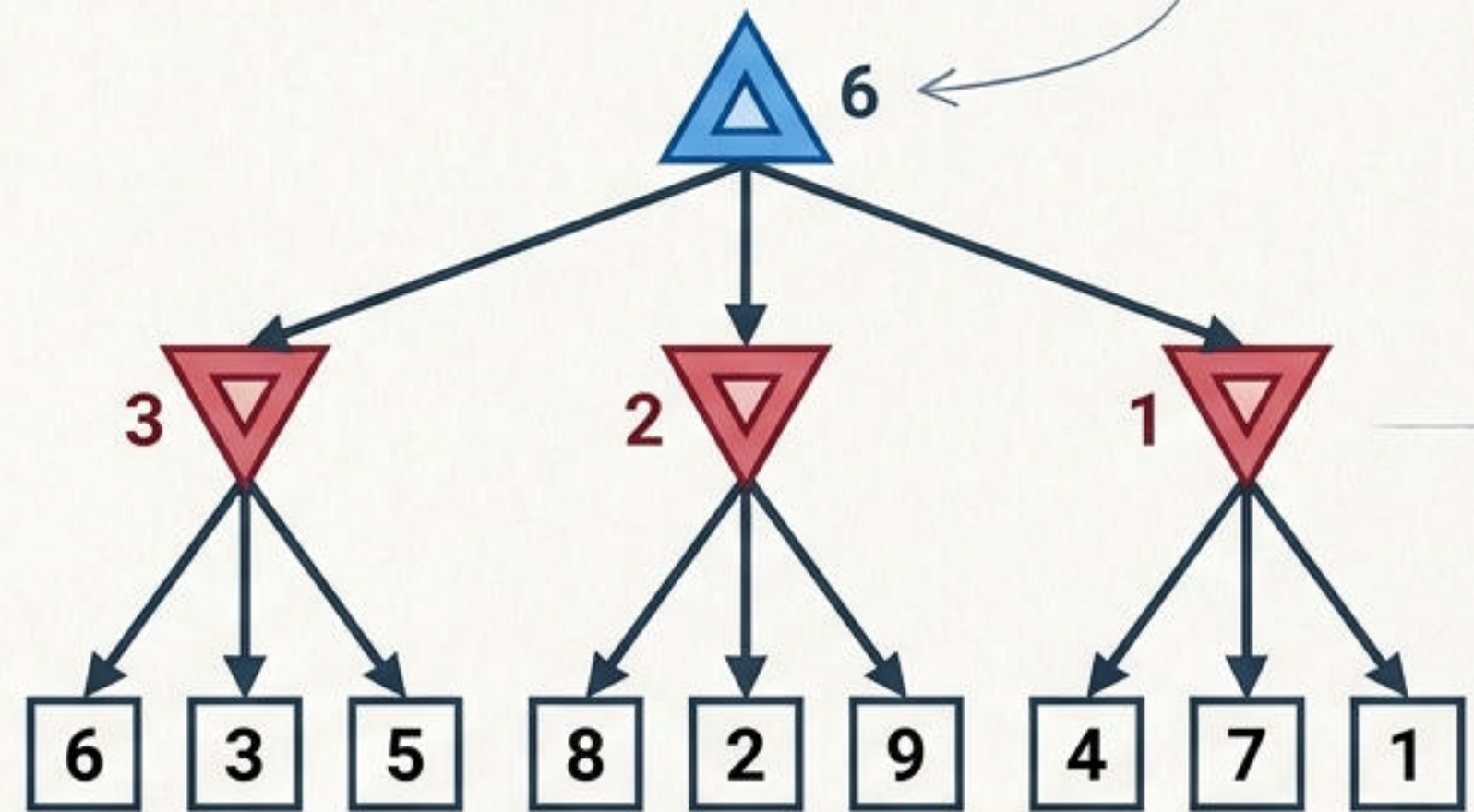


גיזום משפר את זמן הריצה, אך לעולם לא משנה את תוצאת השורש של אלגוריתם המינימקס.

• Max שואף למקסם את הניקוד, בעוד Min שואף למזער אותו.

Standard Minimax

מינימקס סטנדרטי



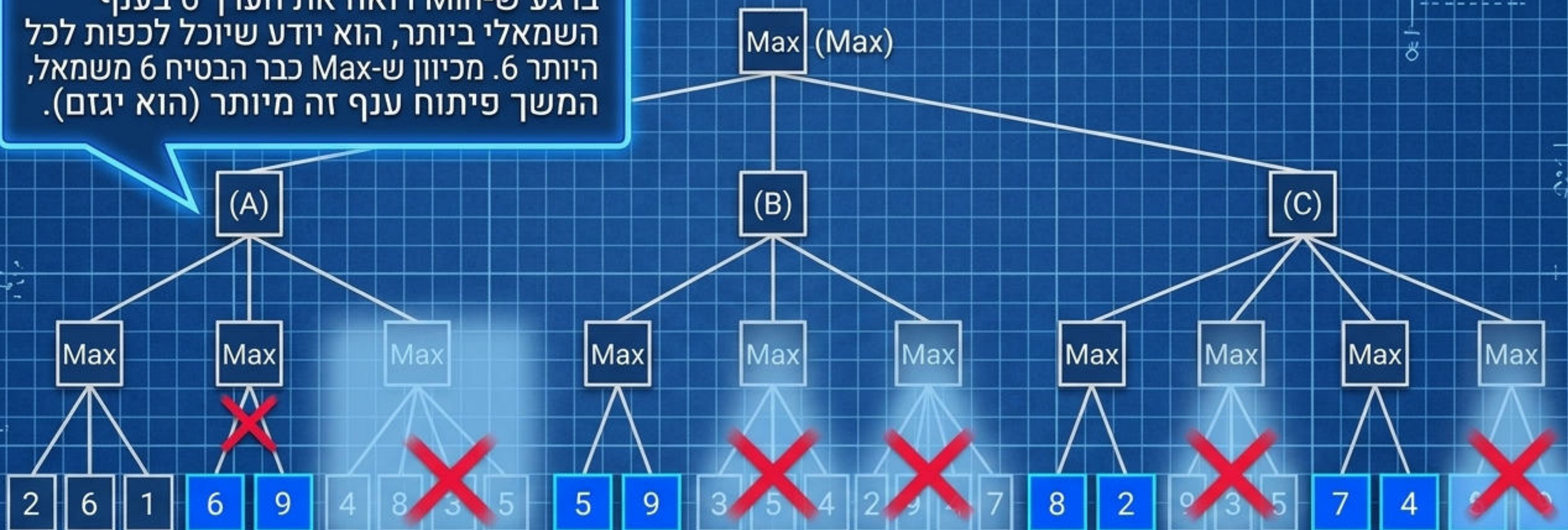
• Max שואף למקסם את הניקוד, בעוד Min שואף למזער אותו.

מיתוסים מול עובדות

הסיבה	פסק דין	הטענה
אלפא-ביתא תמיד מחזיר את אותו ערך לשורש בדיוק בדיוק כמו מינימקס.	 שקר	גיזום מהיר ממינימקס אך מאבד את הבטחת האופטימליות.
צמתים שנגזמו כלל אינם מוערכים ולכן לא מקבלים את הערך המדויק שלהם.	 שקר	הערך שמוחזר זהה למינימקס עבור כל צומת בעץ.
אם היריב עושה טעות, המקס יכול לנצל זאת ולהשיג תוצאה תוצאה טובה מהמובטח.	 אמת	מול יריב תת-אופטימלי, ייתכן ששחקן המקס ישיג ציון גבוה מערך המינימקס $v(s)$.

סריקה משמאל לימין הלכה למעשה

ברגע ש-Min רואה את הערך 6 בענף השמאלי ביותר, הוא יודע שיוכל לכפות לכל היותר 6. מכיוון ש-Max כבר הבטיח 6 משמאל, המשך פיתוח ענף זה מיותר (הוא יגזם).

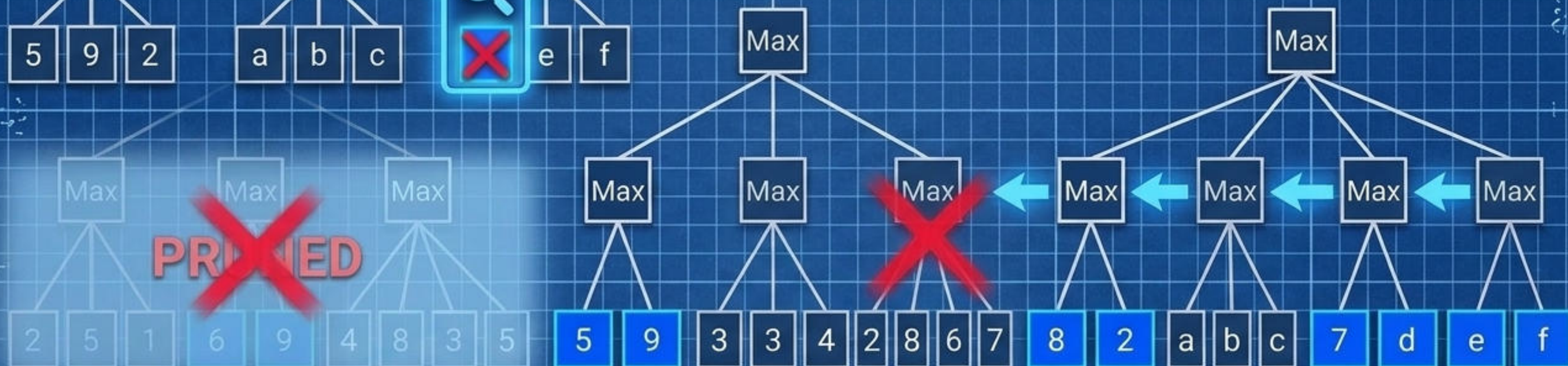
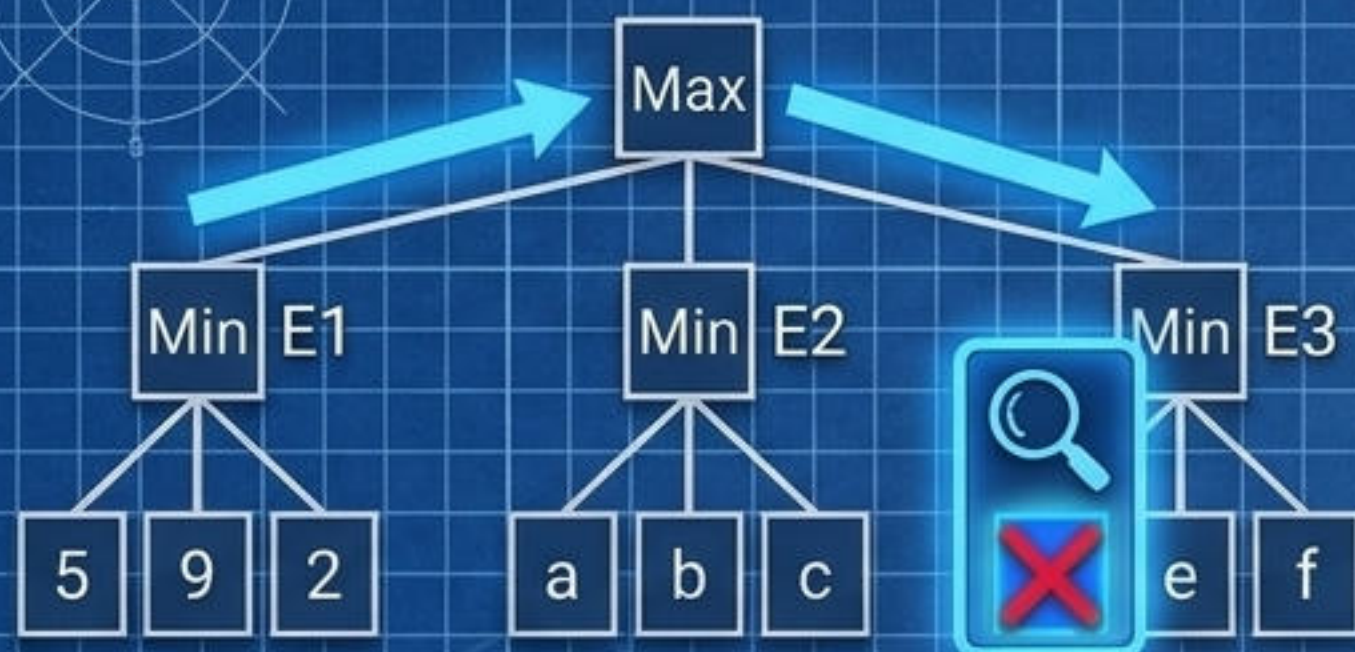


העלים שנבדקו בפועל הם רק 6, 9, 5, 8, 9, 7. כל השאר נגזמו.



סדר הסריקה קובע הכל

בסריקה משמאל לימין: כדי שענף E7 ייגזם, נדרש שערך a יהיה $2 \leq$ (כך ש-Min יבטיח ערך נמוך מספיק כדי ש-Max יתעלם).



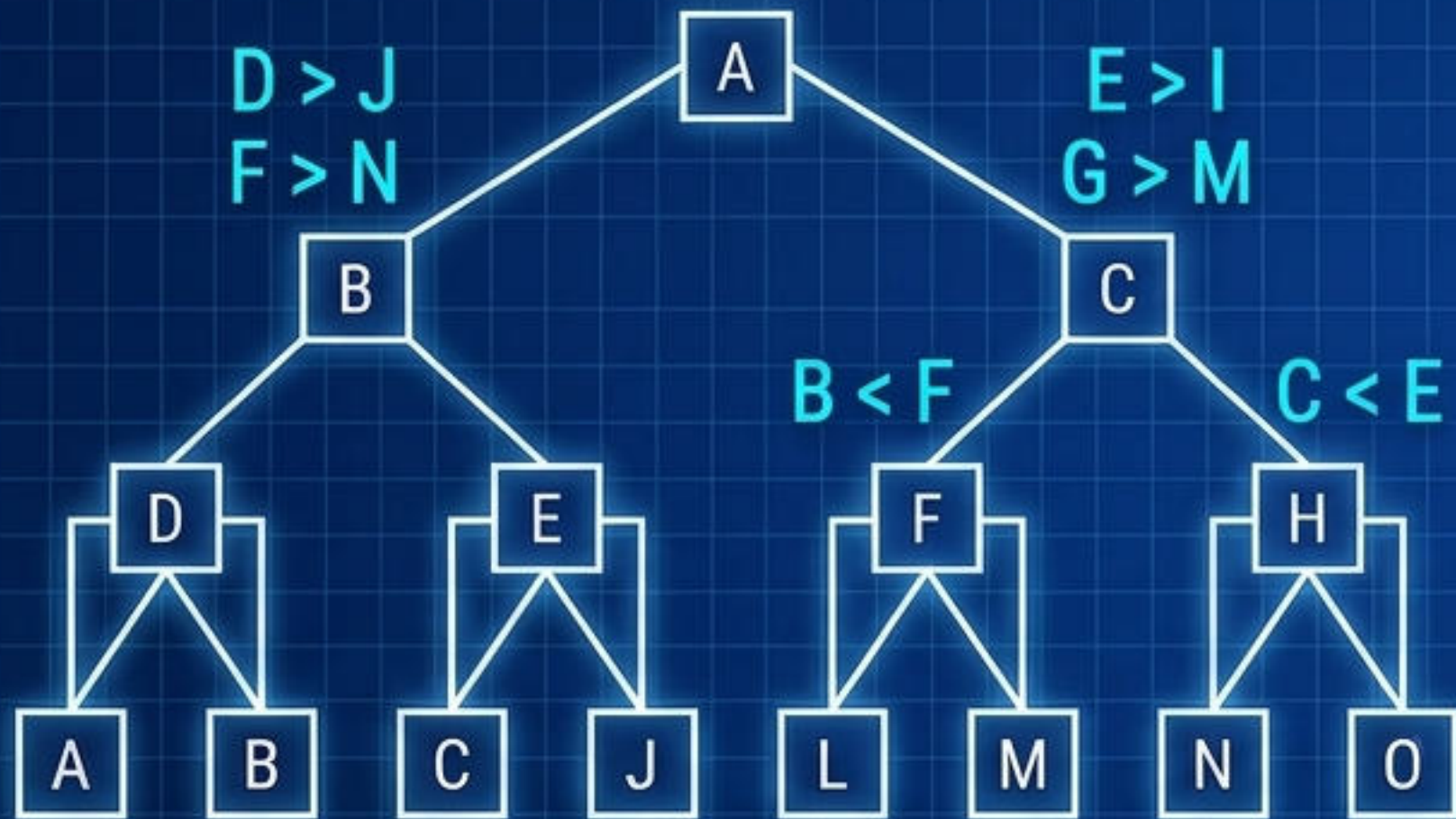
סריקה מימין לשמאל תוביל לגיזום ענפים אחרים לחלוטין (כמו E3 כולו במקרים מסוימים).

ככל שנבדוק את המהלכים הטובים ביותר קודם, כך נגזום יותר ענפים.



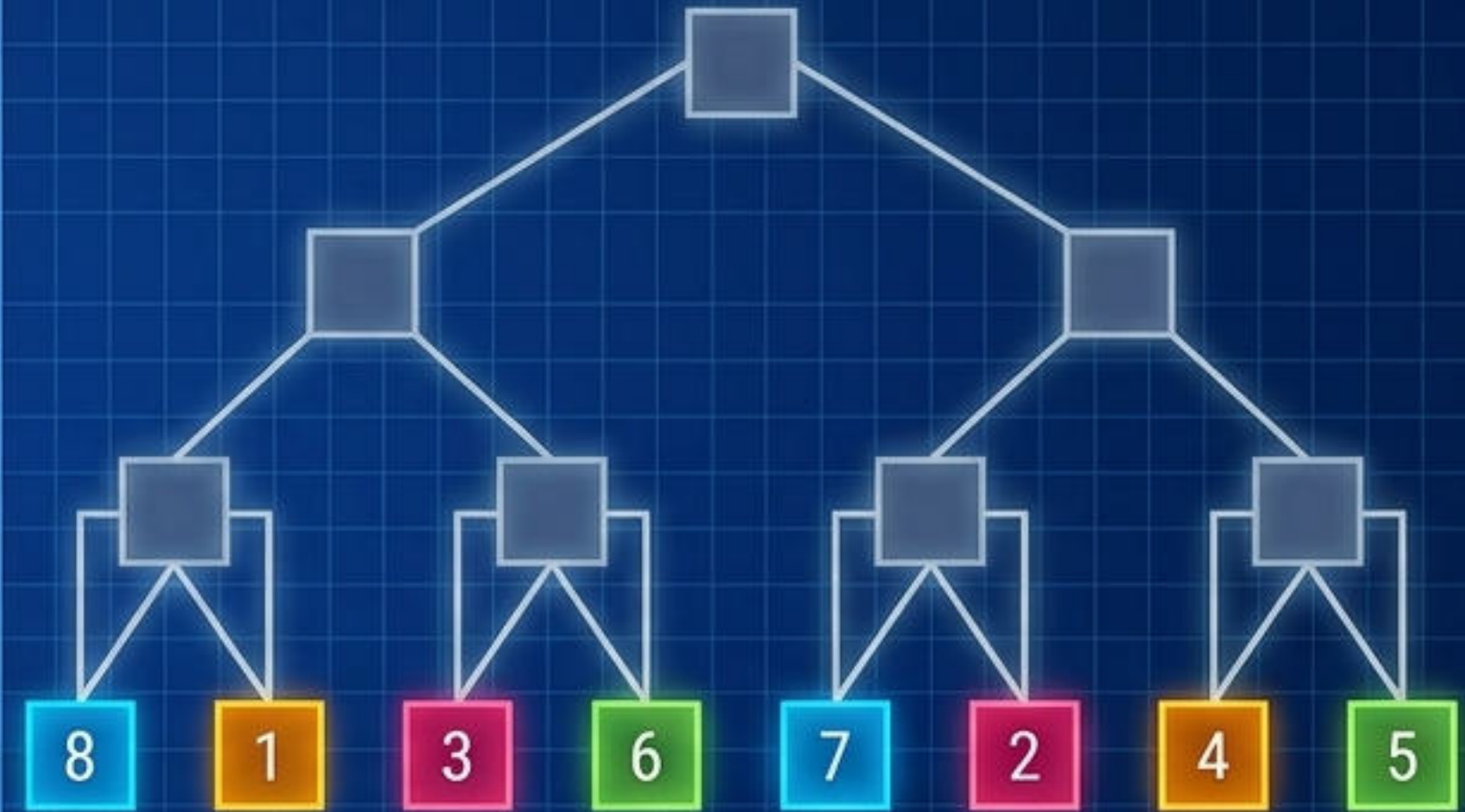
הנדסת העץ: התרחיש הטוב לעומת הגרוע ביותר

The Perfect Tree



מקסימום חיסכון: מתרחש כאשר התנאים (כמו $D > D$) מבטיחים שהענפים הבאים לא ישפרו את המצב.

The Worst-Case Tree



אפס גיזום: בתרחיש הגרוע (כמו בדוגמה), הערכים מסודרים כך שכל ענף חדש מציע פוטנציאל לשיפור, מה שמאלץ את האלגוריתם לסרוק את כל העץ **בדיוק** כמו במינימקס רגיל.

דיבאג מן המציאות: למה תוכנת השחמט קרסה?

```

clast function = {
  state = bin;
  while (upenandFor[i, mehn) {
    if (end == daeu = 0) {
      if (disk) {
        return += #t;
        return new *-watchm + 11;
      }
    }
    else ttimeNowCinIng(dhit.mvs) {
      if (ditur == nev) {
        return new *-watchm + 15;
      }
    }
  }
}

```

לולאה אינסופית: הסטודנט ניסה לפתוח את העץ עד לעלים (מצבי סיום). בשחמט, עומק העץ הוא עצום ולכן האלגוריתם האלגוריתם נתקע. הפתרון: יש להגביל את עומק החיפוש ולהשתמש בפונקציית הערכה.



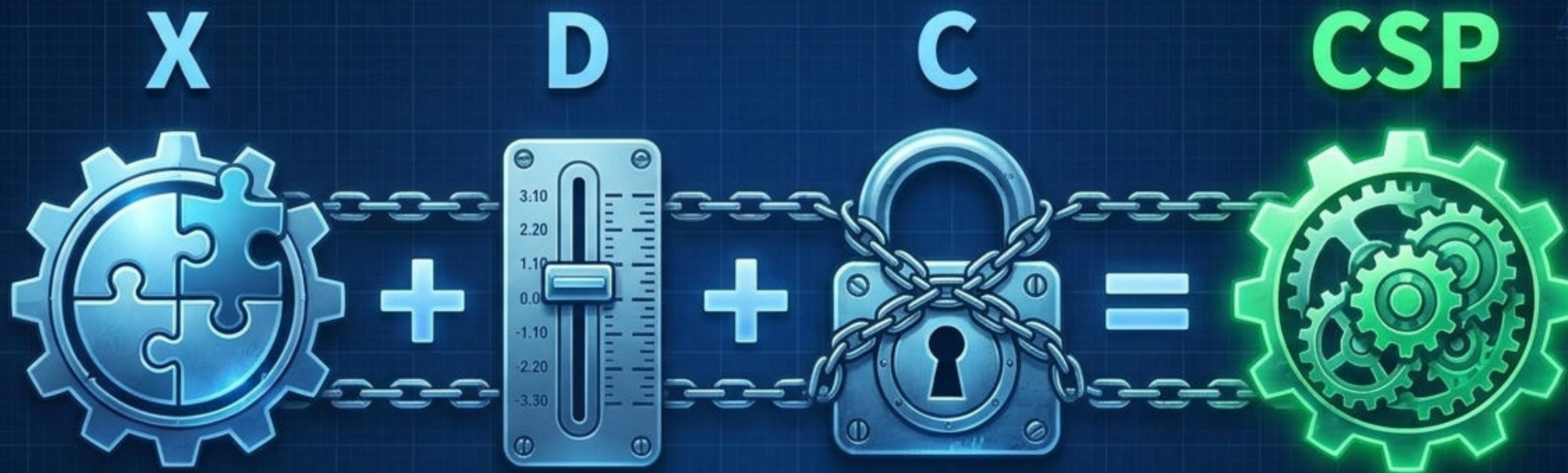
אשליית השלמות של יוריסטיקות: הוספת פונקציית הערכה לא מבטיחה משחק אופטימלי מול יריב מושלם, משום שיוריסטיקה היא רק קירוב ולא חישוב מדויק עד סוף המשחק.



כלל שנבדוק את המהלכים הטובים ביותר קודם, כך נגזום יותר ענפים.



בעיות סיפוק אילוצים (CSPs): האנטומיה של הבעיה



C - אילוצים: החוקים שמגדירים אילו שילובי ערכים מותרים או אסורים בין משתנים שונים.



D - תחומים: טווח הערכים החוקיים שניתן להציב בכל משתנה.



X - משתנים: קבוצת האלמנטים שצריך למצוא עבורם ערך. $(X = \{X_1, X_2, \dots, X_n\})$



מצבי הבעיה: המסע אל הפתרון

שלב 3: השמה שלמה



השמה שלמה: כל
המשתנים קיבלו ערך.

שלב 2: השמה עקבית / חוקית



השמה עקבית / חוקית: השמה
שאינה מפרה אף אילוץ מקבוצת C
(חוק ברזל: תמיד חייב להתקיים).

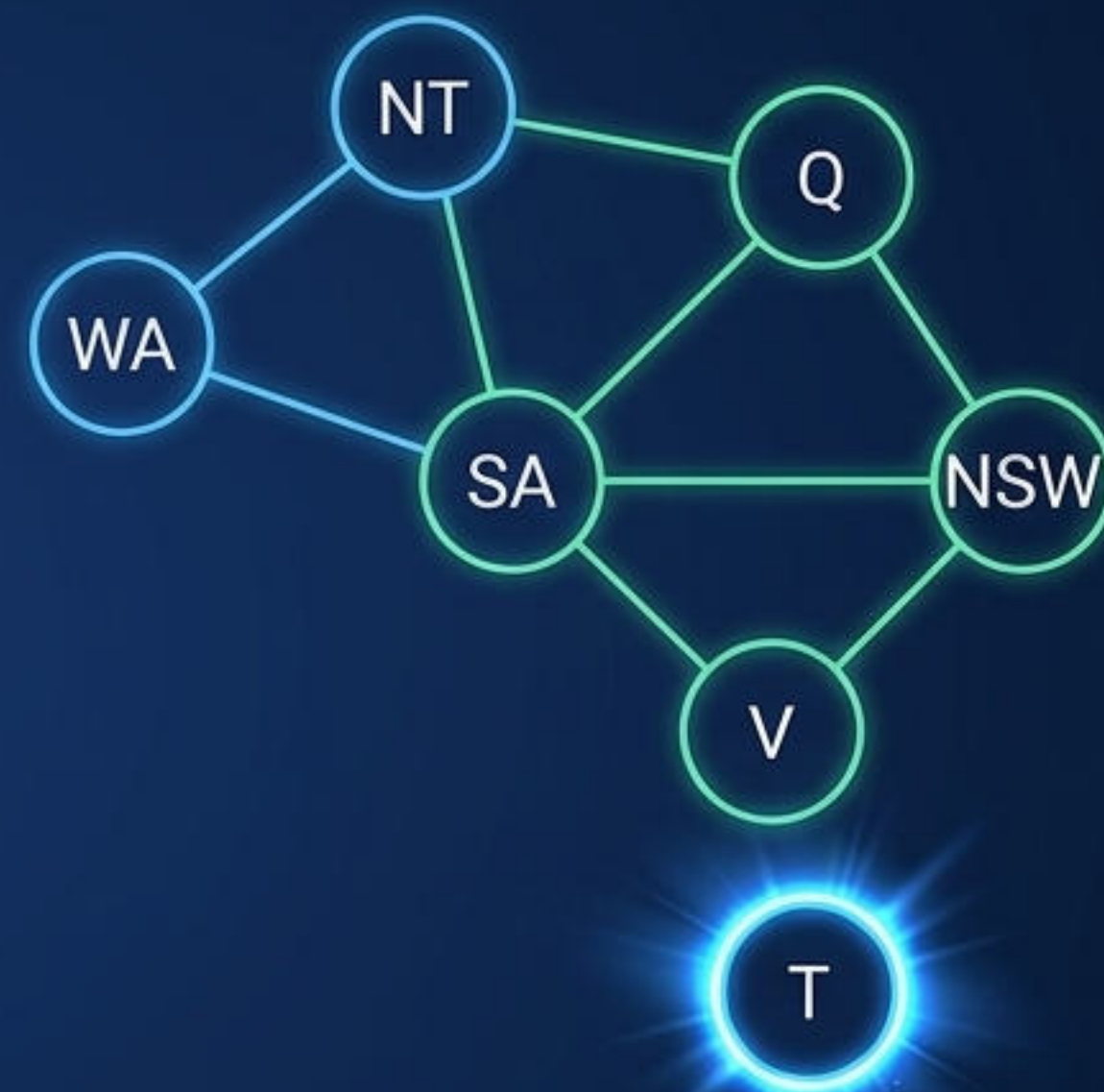
שלב 1: השמה חלקית



השמה חלקית: רק חלק
מהמשתנים קיבלו ערך.

המטרה: פתרון = השמה שלמה + עקבית

גרף אילוצים: כשהגיאומטריה מאיצה חיפוש



גרף אילוצים מעב מתרגם CSP בינארי למפה רשתית. צמתים = משתנים, קשתות = אילוצים.

סוד הקסם (תתי-בעיות): אלגוריתמים משתמשים במבנה הגרף כדי לזהות קיצורי דרך. למשל, טסמניה (T) ממנותקת משאר הגרף, ולכן מהווה תת-בעיה עצמאית. זה חותך את זמן החישוב באופן אקספוננציאלי!



סוגי משתנים ב-CSP



רציף (Continuous)

למשל, תזמון מדויק לטלסקופ
החלל האבל.

אילוצים ליניאריים פתירים
בזמן פולינומי בעזרת שיטות
תכנות ליניארי (LP).



בדיד אינסופי (Discrete Infinite)

למשל, תזמון משימות עם
מספרים שלמים.

דורש שפת אילוצים מתמטית
(למשל: $Start1 + 5 \leq Start3$).

אילוצים ליניאריים פתירים,
לא-ליניאריים אינם כריעים
(Undecidable).



בדיד סופי (Discrete Finite)

תחומים בגודל d .

מספר ההשמות הוא
 $O(d^n)$.

דוגמה: בעיות בוליאניות
(NP-Complete).

סוגי אילוצים: מהפשוט ועד לגלובלי

אילוץ גלובלי / n-ary



מערב מספר רב של משתנים או את כולם יחד. (למשל: חוקי סודוקו שבהם כל הספרות בשורה חייבות להיות שונות - AllDiff).
הערה: ניתן להמיר אילוצים גלובליים לבינאריים באמצעות משתני עזר.

אילוץ בינארי (Binary)



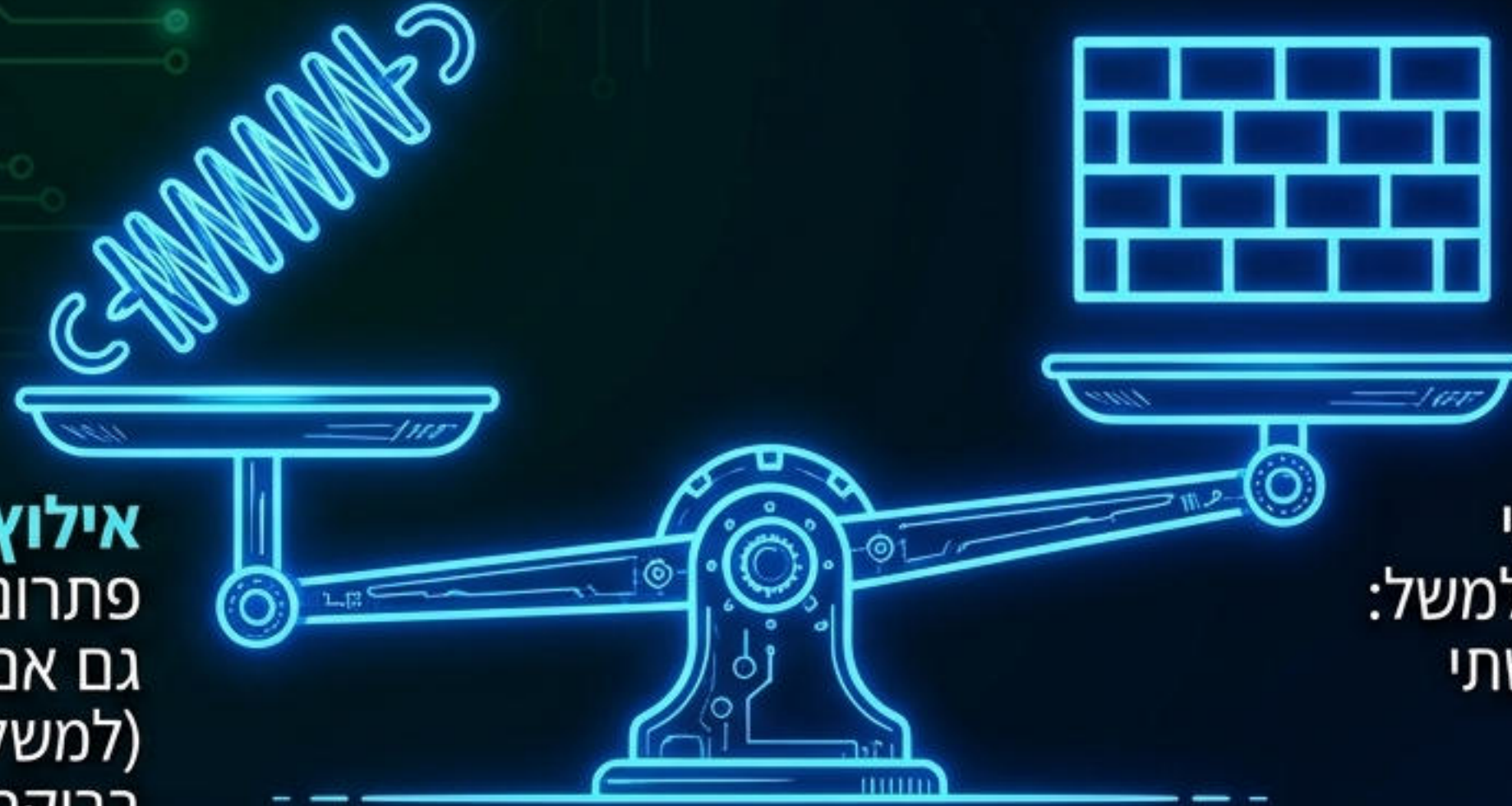
מתייחס ליחס בין שני משתנים. (למשל: $SA \neq NSW$).
זהו הבסיס לגרף האילוצים הסטנדרטי.

אילוץ אונרי (Unary)



מגביל ערך של משתנה יחיד. (למשל: דרום אוסטרליה לא יכולה להיות ירוקה, $SA \neq \text{green}$).

חוקי ברזל מול העדפות: אופטימיזציה של CSP



אילוץ עדיפות: מגדיר אילו פתרונות הינם עדיפים, גם אם אחרים חוקיים. (למשל: פרופ' א' מעדיף ללמד בבוקר).

אילוץ אבסולוטי: תנאי מוחלט שאסור להפר. (למשל: מרצה לא יכול ללמד בשתי כיתות בו-זמנית). הפרה = אין פתרון.

המשמעות התיאורטית: הוספת אילוצי עדיפות הופכת את ה-CSP מחיפוש פתרון בינארי (עובד/לא עובד) לבעיית אופטימיזציה שבה כל הפרת עדיפות גובה 'מחיר'.

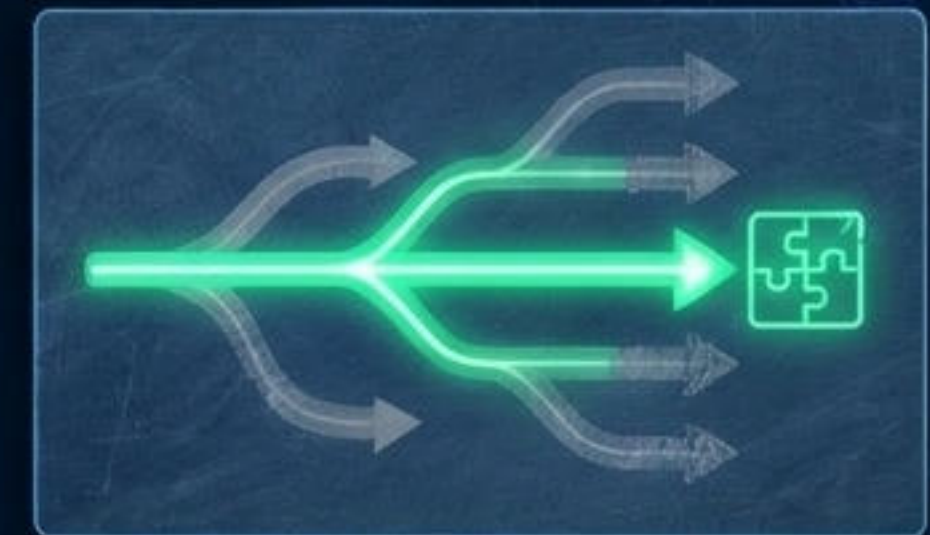
המנוע האלגוריתמי: איך באמת פותרים CSP?



שלב 1: ייצוג הבעיה
הגדרת המשתנים, התחומים והקשרים בגרף.



שלב 2: הפצת אילוצים (Constraint Propagation)
שימוש בחוקים המקומיים כדי לצמצם ולסנן מראש את מרחב האפשרויות (התחומים), לפני שבכלל מתחילים לנחש.



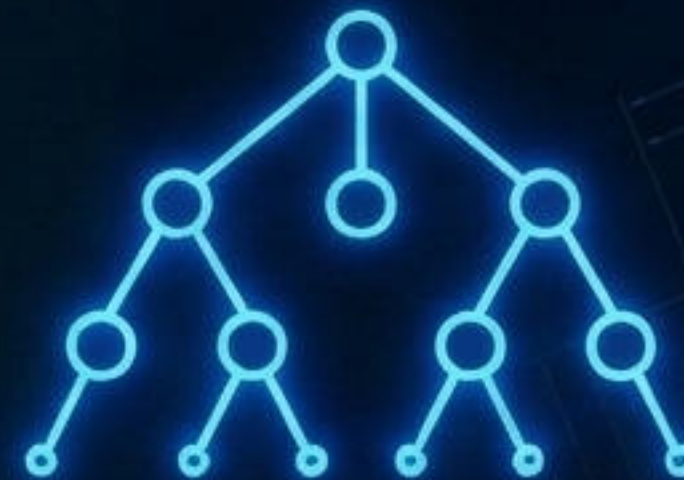
שלב 3: חיפוש מעמיק
שימוש באלגוריתמים כמו Search Backtracking כדי לנווט במרחב הנודר ביעילות, תוך ניצול המבנה הסטנדרטי של הבעיה.

המיקוד לאסטרטג: סיכום עקרונות המפתח



בעיות סיפוק אילוצים (CSP)

- המטרה היא למצוא השמה שהיא גם שלמה וגם עקבית (ללא סתירות).
- מבנה הגרף מגלה סודות: רכיבים מנותקים הם תתי-בעיות שחוסכות זמן רב.
- מודלים רציפים או אילוצי העדפה דורשים כלים מתמטיים מתקדמים (אופטימיזציה).



חיפוש בתנאי יריבות

- גיזום אלפא-ביתא מייעל ריצה אך לעולם אינו משנה את ערך השורש.
- סדר הסריקה קריטי: סריקת המהלכים הטובים ביותר קודם תניב מקסימום גיזום.
- יוריסטיקות הן קירוב בלבד ואינן מבטיחות פתרון מושלם מול יריב אופטימלי.