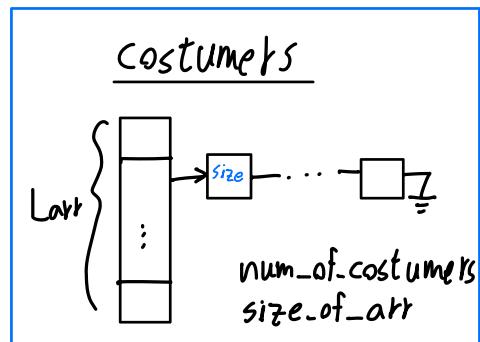


רְפָקֵד נֶהֱרָה שְׁרִוְתָּה גַּעֲמָה
נֶהֱרָה 25. וְאַזְמָן אֶל מִזְבְּחָה תְּנוּנִים וְכֹסְטְּרָם

וְאַזְמָן וְאַזְמָן

רְפָקֵד כְּסֻמְתָּס (1)



כְּסֻמְתָּס אֶל יְהֵא גְּדוֹלָה

בְּכָל "אַזְמָן לְזִבְחָה" כְּהוּ מְרַכְּזִי.

כְּסֻמְתָּס כְּלָשׂוֹן פְּנִיר זָהָב

כְּנַעַץ וְפִזְבָּחָה.

⊗ רְפָקֵד נֶהֱרָה key דְּבָרָה רְפָקֵד נֶהֱרָה (גַּעֲמָה)

כְּנַעַץ וְפִזְבָּחָה (key)

רְפָקֵד נֶהֱרָה בְּגַם הַלְּבָשָׁה

רְפָקֵד id - attrSize

רְפָקֵד ovr - C-num

רְפָקֵד id - key - כְּיָה וְכָל כְּבָדָה

רְפָקֵד id - true - member

רְפָקֵד ovr - phone

רְפָקֵד ovr - c-id

⊗ רְפָקֵד נֶהֱרָה key - B-costumers (3)

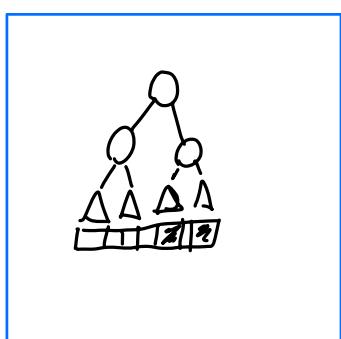
רְפָקֵד id - c_id •

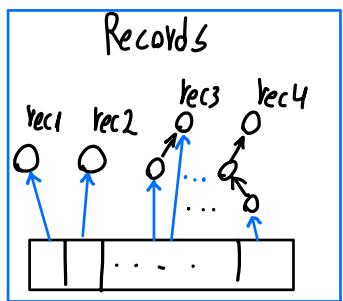
רְפָקֵד height •

רְפָקֵד rank •

רְפָקֵד expence •

רְפָקֵד extra •





4

Union-Find class - records

Hi $\text{rec}[i]$ מאריך מינימום
הנמרג כויה מאריך מינימום
ולאך נזקן מינימום מינימום
לפיה יזנוק records -> קבוצה
שיינט און מינימום
לפיה מינימום - height
ו-
מינימום מינימום - copies
ו-
מינימום מינימום - purchased
ו-
מינימום מינימום - column
ו-
מינימום מינימום - relative

Customer class

Q(1) - customer class constructor נוצרת כפיה ב-init().
כפיה ב-init() נוצרת כפיה ב-customer ו-customer נוצרת כפיה ב-customer.
כפיה ב-customer נוצרת כפיה ב-customer ו-customer נוצרת כפיה ב-customer.
כפיה ב-customer נוצרת כפיה ב-customer ו-customer נוצרת כפיה ב-customer.
Q(1) - customer class constructor נוצרת כפיה ב-customer ו-customer נוצרת כפיה ב-customer.
Q(1) - customer class constructor נוצרת כפיה ב-customer ו-customer נוצרת כפיה ב-customer.
Q(1) - customer class constructor נוצרת כפיה ב-customer ו-customer נוצרת כפיה ב-customer.

~RecordsCompany()

- $\text{records}[i]$ \rightarrow $i \leq i < m$ $\text{rec}(i) \rightarrow$ i תואם ל-
- $\text{records}[i]$ \rightarrow $i \leq i < m$ $\text{costumers} \rightarrow$ i תואם ל-
- $\text{B_customers} \rightarrow$ $i \leq i < m$ $\text{rec}(i) \rightarrow$ i תואם ל-
- $\text{records}[i]$ \rightarrow $i \leq i < m$ $\text{rec}(i) \rightarrow$ i תואם ל-
- $\text{records}[i]$ \rightarrow $i \leq i < m$ $\text{rec}(i) \rightarrow$ i תואם ל-
- $O(n+m)$ סטם $O(n+m)$

Output_t<int> getPhone(int c_id)

(records \rightarrow $i \leq i < m$ $\text{rec}(i) \rightarrow$ i תואם ל-

$\text{records}[i]$ \rightarrow $i \leq i < m$ $\text{costumers} \rightarrow$ i תואם ל-

$\text{costumers}[i]$ \rightarrow $i \leq i < m$ $\text{phone} \rightarrow$ i תואם ל-

$\text{phone}[i]$ \rightarrow $i \leq i < m$ $\text{phone}[i] \rightarrow$ i תואם ל-

StatusType makeMember(int c_id)

- $\text{records}[i]$ \rightarrow $i \leq i < m$ $\text{rec}(i) \rightarrow$ i תואם ל-
- $\text{records}[i]$ \rightarrow $i \leq i < m$ $\text{costumers} \rightarrow$ i תואם ל-
- $\text{costumers}[i]$ \rightarrow $i \leq i < m$ $\text{B_customer} \rightarrow$ i תואם ל-
- $\text{B_customer}[i]$ \rightarrow $i \leq i < m$ $\text{makeMember} \rightarrow$ i תואם ל-
- $O(\log(m))$ סטם $O(\log(m))$
- $O(\log(m))$ סטם $O(\log(m))$
- $O(\log(m))$ סטם $O(\log(m))$
- $O(\log(m))$ סטם $O(\log(m))$

StatusType newMonth(int *records_stocks, int number_of_records)

• תוסף וריאנט של פונקציית records בזאת, גורם ל-1

כדי לא נאיבוד

$\forall 0 \leq i < m$ rec_i^i :

④ רשותה למשוך און-ליין

records מוגדרות מ-0

$O(1) \sim num_of_records$ פונקציית records מקבלת סימולן •

• רשותה למשוך און-ליין על בסיס גודלה. גודלה :

• $0 \leq i < m$ i לא מוגדר

• $height = 0$, $copies = records_stocks[i]$, $purchased = 0$, $column = i$

$O(1) \sim height$ אם $height = 0$ פונקציית records מקבלת סימולן •

• $O(1) \sim copies$ אם $copies = 0$ פונקציית records מקבלת סימולן •

• $O(1) \sim purchased$ אם $purchased = 0$ פונקציית records מקבלת סימולן •

$O(m) \sim column$ פונקציית records מקבלת סימולן •

• $O(n) \sim height$ פונקציית records מקבלת סימולן •

• $O(n) \sim copies$ פונקציית records מקבלת סימולן •

• $O(n) \sim purchased$ פונקציית records מקבלת סימולן •

• $O(n) \sim column$ פונקציית records מקבלת סימולן •

• $O(n)$ \leq $O(n+m)$ \leq $O(2n+m)$ \leq $O(2n+2m)$ \leq $O(n+m+n)$ \leq $O(2n+m)$ \leq $O(n+m)$ \leq $O(n)$

```
StatusType addCostumer(int c_id, int phone)
```

: O(1)-2 c_num alc fpc

: C_num < attrSize pl I

ככ. 6(7), כט נרכזין (100ס) 7, כט (פ2 פג'ב) ו

$\langle f_p \rangle \approx g_{3NN} Q(1) \sim g_{3\omega N} \pi^2 m^2 / 12$

אָחָלָס נְצָרָנִים נַעֲמָנִים לְפָנֵי יְהוָה וְלִפְנֵי כָּל־עַמּוֹת

• סימן מס' 3 – סימן מס' 2 (בנוסף ל-10%)

this → phone = phone

member = false

expences = 0

γαερικοί, γεωνηνοί ο(1) πρώτοι Ι ~8ε1 ο(1) πρώτοι ΙΙΙ ~8ε1

• 1 JUN '01 X3INNS 04122 Y322W (II) 2See 1(k).1

(סנווילר בראון גולדמן ורדרמן וויליאם גולדמן)

• *N*(*z*) *>* *N*(*z*) *if* *N*(*z*) */* *N*(*z*) *>* *N*(*z*)

תְּנִסְתָּוּ רַבְּנוֹתָא וְלִבְּרָהָא

לעתה נזכיר מה שפירושו
 $\sum_{l=1}^{\text{Latr}} C_{\text{add}}$ הוא סכום כל הפעולות שפערת
 $\frac{\text{Latr}}{2}$ מפעלים נתקלים בפערת
 $\log(\gamma_{3NNN})$ $O(1)$ ~ 0.25%

$$C_{\text{add}} = 1 \quad \text{ולא כביכול}$$

$$C_{\text{transfer}} = \text{Latr}$$

: מוגבל ל 1.1%

$$\alpha_{\text{add}} = 3$$

$$\alpha_{\text{transfer}} = 0$$

נזכיר שעבור כל פעולה שפערת $\frac{\text{Latr}}{2}$ מפעלים נתקלים בפערת
 $\log(\gamma_{3NNN})$ מפעלים נתקלים בפערת $\log(\gamma_{3NNN})$, כלומר מילוי פער
 מילוי מוקדם יותר מפערת Latr מילוי מוקדם יותר מפערת Latr
 C_{transfer} ספ

$$\sum_{l=1}^{\frac{\text{Latr}}{2}} C_{\text{add}} < \sum_{l=1}^{\frac{\text{Latr}}{2}} \alpha_{\text{add}} = \text{Latr}$$

. $O(1)$ מזג נזיר פל גיבובן ייון פסוי

. (γ_{3NNN}) γ_{3NNN} $O(1)$ ~ 1/3.2.1N . γ_{3NNN} ~ 0.05

`Output_t<bool> isMember(int c_id)`

γ_{3NNN} $O(1)$ ~ 1/3.2.1N סכום מפעלים נתקלים בפערת $\log(\gamma_{3NNN})$
 מילוי מוקדם יותר מפערת $\log(\gamma_{3NNN})$ מילוי מוקדם יותר מפערת $\log(\gamma_{3NNN})$. (γ_{3NNN}) γ_{3NNN} $O(1)$ ~ 1/3.2.1N member

StatusType buyRecord(int c_id, int r_id)

$\log^*(m) \approx \text{find}(k_{\text{id}})$

• תרשים תכנית מילוי ארון כבאותם

$b(g(n)) \rightarrow B\text{-costumers}$ יחס נזקן וזרום

($100 + t \cdot f(10^t)$) על expence גובה מינימום

Ο.ο. $O(n \log n)$

• תרשים קבוצת ריבוי addPrize

לפניהם קבוצה $\text{add-prize}(\text{rank}, \text{amount})$ קבוצה $\text{add-prize}(\text{rank}, \text{amount})$

האם ניתן לחלק את amount בין rank ?

rank of B-costumers \uparrow דמיון נרחב של addPrize

• נניח ו- x הוא קבוצה נזקנת גלויה:

לפניהם \uparrow מילוי קבוצה נזקנת גלויה \uparrow (I)

extra- N amount מילוי קבוצה נזקנת גלויה \uparrow (II)

• נסחים:

extra- δ amount מילוי קבוצה נזקנת גלויה \uparrow (I)

לפניהם \uparrow מילוי קבוצה נזקנת גלויה \uparrow (II)

• נסחים (I) ו- \uparrow מילוי קבוצה נזקנת גלויה \uparrow (III)

• נסחים (II) ו- \uparrow מילוי קבוצה נזקנת גלויה \uparrow .

• דוגמאות ל- \uparrow מילוי קבוצה נזקנת גלויה.

$O(\log(n))$ כ- \uparrow מילוי קבוצה נזקנת גלויה.

לפניך פוליה רולו

$O(\log(n))$ סדרה, רקורסיבית כ c_id של אדום ונקרא $\text{getRank}(c_id)$
ול rank נקבע בזאת אם $c_id = c_id - 1$ נקבע נובע
 $\tilde{c_id} < c_id - 1$ והוא $\tilde{c_id}$

StatusType addPrize(int c_id1, int c_id2, double amount)

$O(\log(n)) \rightarrow \text{getRank}$ של $c_id1 - 1$ ו- c_id2 לפי הדרישות ונקראם r_1 ו- r_2
ולא ניקח $O(\log(n))$ ונקראם r_2 מ"ג אם c_id2 ונקראם $\text{add-prize}(r_2, amount)$

$\text{add-prize}(r_2 - 1, amount)$ ו- $r_2 - 1$ ניקח

$O(\log(n)) \rightarrow \rho_{\text{rank}} \text{ סדרה}$

$\because O(\log(n)) \rightarrow \text{rank}$ מ"ג אם c_id1 ונקראם r_1
ולא ניקח $\text{add-prize}(r_1, -amount)$ ו- r_1

ולא ניקח $\text{add-prize}(r_1 - 1, -amount)$ ו- $r_1 - 1$ ניקח

$O(\log(n)) \rightarrow \rho_{\text{rank}} \text{ סדרה}$

$O(\log(n)) \rightarrow \rho_{\text{rank}}$ סדרה עליה ניקח c_id1 ונקראם r_1
 $O(\log(n))$ סדרה עליה ניקח c_id2 ונקראם r_2

Output_t<double> geptExpenses(int c_id)

Given c_id and $k3nje$ for $B_costumers$ find for $y1w$
extra- \rightarrow sum \rightarrow sum \rightarrow extra \rightarrow c_id \rightarrow sum \rightarrow $k3nje$

Given c_id and $B_costumers$ for $y1w$ \rightarrow $O(\log(n))$

Given c_id and $k3nje$ for $B_costumers$ find for $y1w$ \rightarrow $O(\log(n))$
 $\max\{0, \text{expenses} - \text{sum}\}$ \rightarrow $O(\log(n))$

$O(\log(n))$ \rightarrow $O(\log(n))$



חברת תקליטים ידועה רוצה ליעל את האופן שבו היא עובדת. לשם כך היא צריכה עזרה במימוש של מבנה נתונים שיתאים לצרכיה.

מטרת המערכת היא לאסוף מידע אודוט הלוקחות והתקליטים שבחנות. לכל לקוחות שנכנסו לחנות יש מזהה ייחודי. שיסומן על ידי c_id (לא יתכן שני לקוחות בעלי אותו c_id) ומספר טלפון. בנוסף, כל לקוח יכול להיות חבר מועדון במידה וירצה בכך. התקליטים שבחנות יופיעו על ידי c_id אשר יהיו ממוספרים באופן רציף מ 0 ועד למספר התקליטים פחות 1, ובכל תחילת חדש מסוף התקליטים ישנה בהתאם לחודש הנוכחי (יתכן כי c_id של התקליט ישנה בין חדשניים) ובנוסף על ידי מספר העותקים ומספר הרכישות שבוצעו עבור תקליט זה. כל לקוח שהינו חבר מועדון אינם משלמים שירות על כל תקליט אלא רק בסוף כל חודש. על מנת לאפשר זאת על המערכת לשמור את סך הרכישות שביצע כל חבר מועדון. בנוסף, על מנת לעודד לקוחות להיות חברי מועדון מידי פעם מתבצעות הגרלות וחלוקת פרסים לחבריו המועדון אשר מזילות את המחיר עליהם שלהם.

לשם כך, עליו למשם מבנה נתונים התומך בפעולות הבאות:
(לאור כל השאלה c הינו מספר הלוקחות במערכת בעוד s הינו מספר התקליטים).

RecordsCompany()

פונקציה בונה של המבנה. על הפונקציה לאותחל את מבנה הנתונים באופן ריק.

פרמטרים: אין.

ערך החזרה: אין.

סיבוכיות זמן: $O(1)$

~RecordsCompany()

פונקציה הורסת של המבנה. על הפונקציה לשחרר ולמחוק את מבנה הנתונים.

פרמטרים: אין.

ערך החזרה: אין.

סיבוכיות זמן: $O(n + m)$

StatusType newMonth(**int** *records_stocks, **int** number_of_records)

הפעולה מאפסת את סידור התקליטים ואת סכום הכספי של כל אדם חי.

ניתן להניח כי פונקציה זו תקרא לפניה כל פעולה הכוללת פעולה עם התקליטים.

פרמטרים: $records_stocks$ – מספר של מספרים המכיל בכל אינדקס את מספר העותקים של תקליט מסוים.
 $number_of_records$ – מספר התקליטים בחודש זה (תואם למספר התאים ב $records_stocks$).

ערך החזרה:

$records_stocks < 0$ – **INVALID_INPUT**

$ALLOCATION_ERROR$ – במידה והייתה שגיאת זיכרון.

SUCCESS – במידה והפעולה הצליחה.

סיבוכיות זמן: $O(n + m)$

StatusType addCostumer(**int** c_id, **int** phone)

הfonקציה מוסיפה לקוח חדש למבנה הנתונים.

פרמטרים: c_id – המזהה של הלוקה, $phone$ – מספר הטלפון שלו.

ערך החזרה:

$phone < 0$ – **INVALID_INPUT**

$ALREADY_EXISTS$ – אם c_id כבר קיים במערכת.

$ALLOCATION_ERROR$ – במידה והייתה שגיאת זיכרון.

SUCCESS – במידה והפעולה הצליחה.

סיבוכיות זמן: $O(1)$ משוערך ממוצע על הקלט.

Output_t<int> getPhone(**int** c_id)

הfonקציה מחזירה את מספר הטלפון של הלוקה.



פרמטרים: c_id – המזהה של הלוקו.

ערך החזרה:

. $c_id < 0$ – INVALID_INPUT

.אם לא קיים לקוחות בעלי c_id – DOESNT_EXISTS

במידה והפעולה הzielich מוחזירה את מספר הטלפון.

סיבוכיות זמן: $O(1)$ בmäßig על הקלט.

StatusType makeMember(int c_id)

הfonקציה הופכת את הלוקו בעל ה c_id לחבר מועדן.

פרמטרים: c_id – המזהה של הלוקו.

ערך החזרה:

. $c_id < 0$ – INVALID_INPUT

.אם לא קיים לקוחות בעלי c_id – DOESNT_EXISTS

.אם הלוקו בעל c_id הוא כבר חבר מועדן – ALREADY_EXISTS

.במידה והייתו שגיאת זיכרון – ALLOCATION_ERROR

.במידה והפעולה הzielich – SUCCESS

סיבוכיות זמן: $O(\log n)$.

Output_t<bool> isMember(int c_id)

הfonקציה מוחזירה האם הלוקו בעל ה c_id הינו חבר מועדן.

פרמטרים: c_id – המזהה של הלוקו.

ערך החזרה:

. $c_id < 0$ – INVALID_INPUT

.אם לא קיים לקוחות בעלי c_id – DOESNT_EXISTS

.במידה והפעולה הzielich – SUCCESS

סיבוכיות זמן: $O(1)$ בmagic על הקלט.

StatusType buyRecord(int c_id, int r_id)

הfonקציה מוסיפה רכישה אחת לתקליט ובמידה והлокו הינו חבר מועדן מוסיפה להוצאות החודשיות שלו את מחיר התקליט שנקבע על ידי הנוסחה הבאה: $t + 100$. כאשר t הינו מספר הרכישות שבוצעו לתקליט זה לפני קניה זו.

פרמטרים: c_id – המזהה של הלוקו, r_id – המזהה של התקליט.

ערך החזרה:

. $r_id < 0$ – INVALID_INPUT

. $r_id \geq number_of_records$ – אם לא קיים לקוחות בעלי c_id או r_id – DOESNT_EXISTS

.במידה והפעולה הzielich – SUCCESS

סיבוכיות זמן: $O(\log n)$.

StatusType addPrize(int c_id1, int c_id2, double amount)

הfonקציה מוסיפה פרט לחבר המועדן בעלי c_id המקיימים $c_id1 \leq c_id < c_id2$ ("יתכן כי c_id1, c_id2 אינם מזהים הנמצאים במערכת"). הפרט הינו בגודל $amount$ והוא יורד מההוצאות החודשיות של כל מי שזכה בו.

פרמטרים: c_id1, c_id2 – טווח מזהים הזוכים, $amount$ – גודל הפרט.

ערך החזרה:

. $amount \leq 0$ – $c_id2 < c_id1, c_id1 < 0$ – INVALID_INPUT

.במידה והפעולה הzielich – SUCCESS

סיבוכיות זמן: $O(\log n)$.

Output_t<double> geptExpenses(int c_id)

הfonקציה מוחזירה את סך הרוצאות החודשיות של חבר המועדן בעל c_id (הרוצאות כוללות גם את הרוזנות בעקבות הפרסים).

פרמטרים: c_id – המזהה של הלוקו.

ערך החזרה:



$c_id < 0 - INVALID_INPUT$

$DOESNT_EXISTS$ – אם לא קיים חבר מועדף בעל c_id .

במידה והפעולה הצליחה, מוחזר סך ההוצאות של הלוקוט.

סיבוכיות זמן: $O(n \log n)$

StatusType `putOnTop(int r_id1, int r_id2)`

הfonקציה עורמת את הערמה שבה נמצא התקליט r_id1 על הערמה שבסה נמצא r_id2 . כאשר הגובה של כל תקליט בערמה של r_id1 גדול בגובה של הערמה של r_id2 והעומدة שבה נמצאים היא באינדקס של האיבר התיכון ביותר בערמה של r_id2 .

פרמטרים: r_id1, r_id2 – המזהים של תקליטים.

ערך החזרה:

$r_id1, r_id2 < 0 - INVALID_INPUT$

$r_id1, r_id2 \geq number_of_records$ – אם $DOESNT_EXISTS$

$r_id1 = r_id2$ – אם אחד מעלה השני.

$r_id1 > r_id2$ – $FAILURE$

$r_id1 < r_id2$ – $SUCCESS$

סיבוכיות זמן: $O(\log^* m)$

StatusType `getPlace(int r_id, int *column, int *height)`

הfonקציה מחזירה את העומדה והגובה של התקליט r_id במשתנים $column, height$ בהתאם.

פרמטרים: r_id – המזהה של תקליט, $column, height$ – משתנים שייכלו את ערכי החזרה.

ערך החזרה:

$column, height = NULL$ – $r_id < 0 - INVALID_INPUT$

$r_id \geq number_of_records$ – אם $DOESNT_EXISTS$

$r_id < number_of_records$ – $SUCCESS$

סיבוכיות זמן: $O(\log^* m)$

הנחהיות:

חלק ייש:

- החלק היבש הוא חלק מהציגן על התרגיל כי שמצוין בנהלי הקורס.
- לפני מימוש הפעולות בקוד יש לתכנן היטב את מבני הנתונים והאלגוריתמים ולוזוד כי אפשרותם למש את הפעולות בדרישות הזמן והזיכרון שלעיל.
- הגשת החלק הרטוב מהווה תנאי הכרחי לקבלת ציון על החלק היבש, כמובן, הגשה בה יתקבל אך ורק חלק ייבש תגרור ציון 0 על התרגיל כולו.
- יש להכין מסמך הכלול תיאור של מבני הנתונים והאלגוריתמים בהם השתמשם ביצירוף הוכחת סיבוכיות הזמן והמקום שלהם. חלק זה עומד בפני עצמו וצריך להיות מובן לפחות לפני העיון בקוד. אין צורך לתאר את הקוד ברמת המשתנים, הfonקציות והמחלקות, אלא ברמה העקרונית. חלק ייבש זה לא תיעוד קוד.
- ראשית הציגו את מבני הנתונים בהם השתמשתם. רצוי ומומלץ להיעזר בצייר.
- לאחר מכן הסבירו כיצד מימושם כל אחת מהפעולות הנדרשות. הוכיחו את דרישות סיבוכיות הזמן של כל פעולה תוך כדי התייחסות לשינויים שהפעולות גורמות מבני הנתונים.
- הוכיחו שמבנה הנתונים וכל הפעולות עומדים בדרישת סיבוכיות המקום.
- החסמים הנתונים בתרגיל הם לא בהכרח הדוקים ולכן יכול להיות שגם שקיים פתרון בסיבוכיות טוביה יותר. מספיק להוכיח את החסמים הדרושים בתרגיל.
- רמת פירוט: יש להסביר את כל הפרטים שאינם טריויאליים ושחשובים לצורך שימוש הפעולות ועמידה בדרישות הסיבוכיות. אין לדון בפרטים טריויאליים (הפעילו את שיקול דעתכם בקשר לזה), ושאלו את האחראי על התרגיל אם איןכם בטוחים). אין לאטט קטעים מהקוד כתחליף להסביר. אין צורך לפרט אלגוריתמים שנלמדו בכתה. כמו כן, אין צורך להוכיח תוצאות ידועות שנלמדו בכתה, אלא מספיק לציין בבירור לאיזו תוצאה אתם מתכוונים.
- על חלק זה לא לצאת מ-8 עמודים.
- והci חשוב **keep it simple!**



חלק רטוב:

- מומלץ למשתמש תחילת את מבני הנתונים באוצר הכללית ביותר ורק אז למשתמש בהונקציות הנדרשות בתרגיל.
- אנו ממליצים בחום על שימוש **Object Oriented**, **C++**, מימוש כזה יאפשר לכם להגיע לפתרון פשוט וקצר יותר לפונקציות אותן עלייכם למשתמש ויאפשר לכם להקליל בקלות את מבני הנתונים שלכם (Ձרו שיש תרגיל רטוב נוספת בהמשך הסמסטר).
- על הקוד להתקומפל על 3 cs1 באופן הבא:
g++ -std=c++11 -DNDEBUG -Wall -c mainWet2.cpp
עליכם מוטלת האחוריות לוודא קומפקצייה של התכנית ב+ + g. אם בחרתם לעובד בקומפיילר אחר, מומלץ לקומפל ב+ g מיד פעם במהלך העבודה.



הערות נוספת:

- כתימות הונקציות שעלייכם למשתמש ומספר הגדרות נמצאים בקובץ **h.recordsCompany**.
- קראו היטב את הקובץ הב"ל, לפני תחילת העבודה.
- אין לשנות את הקבצים **mainWet2.cpp** ו- **mainWet2.h** אשר סופקו כחלק מהתרגיל, ואין להציג אותם.
 - את שאר הקבצים ניתן לשנות.
 - תוכלו להוסף קבצים נוספים, ולחגש אותם.
 - העיקר הוא שהקוד שאותם מגישים יתקمل עמו הפוקודה לעיל, כאשר מוסיףם לו את שני הקבצים **utilesWet2.cpp** ו- **utilesWet2.h**.
- עליכם למשתמש בעצמכם את כל מבני הנתונים (למשל אין **להשתמש במבנים של STL** או אין להוריד מבני נתונים מהאינטרנט).
- **חלק מטהlixir** הבדיקה אנו נבעצ' בדיקה ידנית של הקוד ונווודא שאכן מימשתם את מבני הנתונים שבהם השתמשתם.
- בפרט, אסור להשתמש ב-**pair**, **vector**, **iterator**, **std::pair**, **std::vector**, או כל אלגוריתם של **STL**.
- ניתן להשתמש במצביים חכמים (shared pointers) Smart pointers (כמו **_ptr**), בספריית **math** כמו **exception**.
- חשוב לוודא שאתם מקצים/משחררים זיכרון בצורה נכון (מושלץ לוודא עם **valgrind**). לא חייבים לעבוד עם מצביים חכמים, אך אם אתם מחליטים כן לעשות זאת, לוודא שאתם משתמשים בהם נכון. (תזכרו שהם לא פתרון קסם, למשל, כאשר יוצרים מעגל בהצבעות)
- שגיאות של **ALLOCATION_ERROR** בד"כ מעידות על זליגה בזיכרון.
- מצורפים לתרגיל קבוע קלט ופלט לדוגמא, ניתן להריץ את התוכנה על הקלט ולהשווות עם הפלט המצויר.
- **שים לב:** התוכנית שלכם תבדק על קלטים שונים מקבצי הדוגמא הב"ל, שהיינו ארוכים ויכללו מקרים קצה שונים. לכן, מומלץ מאוד ליצור בעצמכם קבוע קלט, לבדוק את התוכנית עליהם, ולוודא שהיא מטפלת נכון בכל מקרה הקצה.

הגשה:

חלק י Bush + חלק רטוב:

הגשת התרגיל הנה **אר ובק**ALKTRONIT דרכ' אתר הקורס.

יש להגיש קובץ **ZIP** שמכיל את הדברים הבאים:

- בתיקייה הראשית:
 - קבצי ה-**Source Files** שלכם. **למעט הקבצים mainWet2.cpp ו- mainWet2.h**, שאסור לשנות.



- קובץ PDF בשם pdf.dry אשר מכיל את הפתרון הביש. מומלץ לה拷יד את החלק זהה אך ניתן להציג קובץ PDF מבוסס על סריקה של פתרון כתוב בכתב יד. שימו לב כי במקרה של כתוב לא קרייא, כל החלק השני לא תיבדק.
- קובץ txt, submissions.hos, המכיל בשורה הראשונה את שם, תעודה זהה וכתובת הדוא"ל של השופט הראשון ובשורה השנייה את שם, תעודה זהה וכתובת הדוא"ל של השופט השני.
לדוגמה:

John Doe 012345678 doe@cs.technion.ac.il
Henry Taub 123456789 taub@cs.technion.ac.il

■ שים לב כי אתם מגישים את כל שלושת החלקים הנ"ל.

- אין להשתמש בפורמט כיווץ אחר (לדוגמה RAR), לאחר ומערך הבדיקה האוטומטי אינו יודע לזיהות פורמטים אחרים.
- יש לוודא שכאר נכנים לקובץ הדיפ הקבצים מופיעים מיד בתוכו ולא בתוך תיקיה שבתוכו קובץ הדיפ. עברו הgesha שבה הקבצים יהיו בתוך תיקייה, הבדיקה האוטומטית לא תמצא את הקבצים ולא תוכל ל�מפל ולהריץ את הקוד שלהם וכן תיתן אוטומטית 0.
- לאחר שהगשתם, יש באפשרותכם לשנות את התוכנית ולהגיש שוב. ההgesha الأخيرة היא הנחשבת.
- אחרי שהגשתם, אם לא תעמוד בקריטריונים הנ"ל תפסל ותקנס בנקודות!
הgesha שלא תעמוד בקריטריונים הנ"ל מומלץ מאוד לשרת ולהריץ את הבדיקות שלהם עליה כדי לוודא שתם מגישים את הקוד שהתכוונתם להגיש בדיק (ושהוא מתקמפל).

דוחות ואיתורים בהגשה:

- דוחות בתרגיל הבית תיננתנה אך ורק לפי תקנון הקורס.
- 5 נקודות יורדו על כל יום איתור בהgesha ללא אישור מראש. באפשרותם להגיש תרגיל באיחור של עד 5 ימים לפחות אישור. תרגיל שיוגש באיחור של יותר מ-5 ימים ללא אישור מראש יקבל 0.
- במקרה של איתור בהgesha התרגיל יש עדין להגיש את התרגיל אלקטטרונית דרך אתר הקורס.
- בנסיבות להgesha מאוחרת יש להפנות למתרגל האחראי בלבד בכתובת turutovsally@campus.technion.ac.il. לאחר קבלת אישור במיל על הgesha, מספר הימים שאושרו לכם נשמר אצלנו. לכן, אין צורך לצרף להgesha התרגיל אישורים נוספים או את שער ההgesha באיחור.

בצלחה!