<u>מבני נתונים 1</u>

<u>גליון (רטוב) 2 (החלק היבש)</u>

<u>מגישים:</u>

מייל	ת.ז	שם
eilon.halevy@campus.technion.ac.il	328137831	אילון הלוי
maximgurwitz@campus.technion.ac.il	322207671	מקסים גורביץ

<u>דרישות מבנה הנתונים:</u>

אתחול מבנה נתונים ריק. ללא ספינות (מקרה הגרוע (m + m) (מקרה הגרוע (m + m) (מוקם צי חדש בגל הזיכרון. מוקם צי חדש בעל מזהה fleetld (מוקרה הגרוע (מוקרה הגרוע (מוקרה הגרוע (מוקרה הגרוע (מוקרה הגרוע (מוקרה לצי יש ספינה אחת ואין לו משוערך בממוצע (מוקרה מוקרט (מוקרה מוקרט (מוקרה מוקרט (מוקרה מוקרט (מוקרט (מ			
$0(n+m)$ במקרה הגרוע במנו הנתונים וכל הזיכרון. $O(1)$ (fleetId מוקם צי חדש בעל מזהה fleetId מוקט במוצע (חל הלצ' ש ספינה אחת ואין לו משוערך במוצע (חל הלצ' ש ספינה אחת ואין לו משוערך במוצע (חל הלצ' ש ספינה אחת ואין לו משוערך במוצע (חל הקלט משוערך במוצע (חל הקלט מוערן במוצע (חל הקלט הוא יקבל (חל הקלט מוערן במוצע (חל הקלט בספינות בצ' (חל הקלט מוערן במוצע (חל הפיות בצי של שווין המזהה החדש הוא מכם מפר הספינות בצי של שווין המזהה החדש הוא מכם מפר הספינות בצי מוער הבמוצע (חל הפיות בצי מוער הביים, והדרגה של ביל המאודה של הצי הקטן יותר הגדל בשני הציים, והדרגה של ביל הוסיף צי חדש עם מזהה זה. **StatusType unite_fleets (int fleetId1, int fleetId2)** **StatusType pirate_argument (int pirateId1, int pirateId2)** **StatusType pirate_argument (int pirateId1, int fleetId2)** **StatusType pirate_argument (int pirateId2)** **StatusType add_pirate(int pirateId1, int pirateId2)** **StatusType add_pirate(int pirateId)** **StatusType add_pirate(int pirateId1, int pirateId2)** **StatusType add_pirate(int pirateId int pirateId2)** **StatusType unite_fleets(int fleetId)** **StatusType argument (int pirateId int pirat	0(1)	אתחול מבנה נתונים ריק. ללא ספינות	oceans_t()
אר במקרה הגרוע (אוקר מוקר מוקר מוקר מוקר מוקר מוקר מוקר מ			
אר הארולה לצי יש ספינה אחת ואין לו משוערך בממוצע (mr grateld מרותלה לצי יש ספינה אחת ואין לו משוערך בממוצע (mr grateld מרותלה לצי יש ספינה אחת ואין לו על הקלט (mr grateld מרוים מחיל בלי משוערך בממוצע (mr grateld מרוים מחיל בלי משוערך בממוצע (mr grateld מרוים מחיל בלי מחיל בלי (mr grateld מרוים מחיל מחיל מחיל מחיל מחיל מחיל מחיל מחיל	O(n+m)	שחרור מבנה הנתונים וכל הזיכרון.	virtual ~oceans_t()
בהתחלה לצי יש ספינה אחת ואין לו שוערך בממוצע (מ'הקלט (log* m) (פ'ראטים. בעל מדהה pirateld מגוייס (פ'ראטים בעל מדהה pirateld מגוייס (מ'חלט (מ'	במקרה הגרוע		
פיראטים. פיראטים מגוייס (אר מיקלט (מין m) (מרונה pirateld מגוייס (מרונה משוערך בממוצע (אר מין מגוייס (מרונה משוערך בממוצע (אר מין מגוייס (מרונה משוערך בממוצע (מרונה בין אין מרונה בין אין מרונה משוערך בממוצע (מרונה בין מרונה משוערך בממוצע (מרונה משוער ביי משוער בממוצע (מרונה מונה מונה מונה מונה מותר מספר הפיראט משוערך בממוצע (מרונמונה מונמונה יותר ישלם לפיראט (מרונה מונמונה יותר יטלם לפיראט (מרונה מונה מונה יותר יטלם לפיראט (מרונה מונה מונה יותר יטלם לפיראט (מרונה יותר יטלם לפיראט (מרונה מונה יותר יטלם לפיראט (מרונה יותר יטלם לפיראט (מרונה יותר יטלם לפיראט (מרונה מונה מונה יותר יטלם לפיראט (מרונה מונה מונה מונה מונה מונה מונה מונה מ	0(1)		StatusType add_fleet(int fleetId)
איני איני איני איני איני איני איני איני	משוערך בממוצע	בהתחלה לצי יש ספינה אחת ואין לו	
את הדרגה ללי משוערך בממוצע x משוערך בממוצע x משוערך בממוצע x משוערך בממוצע x מווערך בממוצע x מחזירה את מספר הסיבות בצי x מחזירה את מספר הסיבות בצי x משוערך בממוצע x מחזירה את מספר המטבעות של x משוערך בממוצע x מחזירה את מספר המטבעות של x משוערך בממוצע x מווערך בממוצע x מווערך בממוצע x מווערך בממוצע x מווערך בממוצע x מווער x מווערך בממוצע x מווערך בממוצע x מווערך בממוצע x מווער x מווערך בממוצע x מווערך בממוצע x מווערך בממוצע x מווער x מווערך בממוצע x מווערך בממוצע x מווערך בממוצע x מווער x מווערך בממוצע x מווערך בממוצע x מווערך בממוצע x מווער x מווערך בממוצע x מווערך בממוצע x מווערך בממוצע x מווער x מווערך בממוצע x	על הקלט	.פיראטים	
את הדרגה $x \times x$ פיראטים הוא יקבל (חלקלט $x \times y$ הקלט (חלקלט $x \times y$ הדרגה וא יקבל (חלקלט $x \times y$ הדרגה וא הדרגה (חלקלט $x \times y$ המוצע על הקלט (חלקלט $x \times y$ המוצע על הקלט (חלקלט $x \times y$ האר (חלקלט $x \times$	$O(log^* m)$	מגוייס pirateId פיראט בעל מזהה	StatusType add_pirate(int pirateId, int fleetId)
את הדרגה $x+1$ את הדרגה y מולום את הדרגה y מולום אות הדרגה y מולום אות הדרגה y מולום אות הפור במוצע על הקלט מחזירה את מספר הפינות בצי y מחזירה את מספר המיבעות של y מולון במוצע על הקלט y מחזירה את מספר המיבעות של y מולון במוצע על הקלט y מחזירה את מספר המיבעות של y מונוערן בממוצע על הקלט y מונוערן בממוצע על הקלט y	משוערך בממוצע	לצי $fleetId$. הפיראט מתחיל בלי	
0(1)	על הקלט	כסף ואם יש בצי <i>x</i> פיראטים הוא יקבל	
0(1)		x+1 את הדרגה	
על הקלט salary בסך salary מחזירה את מספר הספינות בצי מחזירה את מספר הספינות בצי מחזירה את מספר הספינות בצי הווערן במוצע להקלט output_t < int > num_ships_for_fleet(int fleetId) ס מחזירה את מספר המטבעות של output_t < int > num_ships_for_fleet(int fleetId) ס מחזירה את מספר המטבעות של output_t < int > get_pirate_money(int pirateId) ס מחזירה את מספר המטבעות של output_t < int > get_pirate_money(int pirateId) ס (log* m) איחוד הציים בעלי המזהה מחדש שלהם משוערך בממוצע של הקלט מבחינת מספר הפיראטים (במקרה המאוחד הוא סכום מספר הספינות בצי של שוויון המזהה החדש הוא מפר הספינות בצי של שוויון המזהה של כל הפיראטים בצי הקטן יותר תגדל הפיראטים בצי הקטן יותר תגדל המזהה של הצי הקטן, ולא ניתן לאחר האיחוד לא קיים יותר צי עם ממספר הפיראטים בצי הגדול יותר. המזהה של הצי הקטן, ולא ניתן לאחר האיחוד לא קיים יותר צי עם מזהה זה. $O(log* m)$ וויכוח בין שני פיראטים. הפיראט משוערך בממוצע של הקלט שדרגתו נמוכה יותר ישלם לפיראט מושוערך בממוצע של הקלט שדרגתו נמוכה יותר. סכום הכסף על הקלט שדרגתו נמוכה יותר. סכום הכסף על הקלט מדרגתו נמוכה יותר. סכום הכסף על הקלט מדרגתו נמוכה יותר. סכום הכסף על הקלט שדרגתו נמוכה יותר. סכום הכסף על הקלט שדרגתו נמוכה יותר. סכום הכסף על הקלט	0(1)		StatusType pay_pirate(int pirateId, int salary)
$O(log^*m)$ מחזירה את מספר הספינות בצי. fleetId output_t < int > num_ships_for_fleet(int fleetId) output_t < int > num_ships_for_fleet(int fleetId) output_t < int > num_ships_for_fleet(int fleetId) output_t < int > pirateId and into pirateId and pirateId and pirateId and pirateId and pirateId and pirateId and pirateId pirateId and pirateId output_t < int > get_pirate_money(int pirateId) output_t < int > get_pirate_money(int pirateId) output_t < int > get_pirate_money(int pirateId) output_t < int > feetbus up int into pirateId output_t < int > get_pirate_money(int pirateId) output_t < int > get_pirate_money(int pirateId) output_t < int > get_pirate_money(int pirateId) output_t < int > feetbus up into pirateId > feetletId feetId		מטבעות. salary בסך	
על הקלט המוצר מוצע הפר המטבעות של הקלט משוערך בממוצע אל הקלט (1) (2) (2) (2) (2) (2) (3) (3) (4) (4) (4) (4) (4) (4) (4) (4) (4) (4			output t < int > num ships for fleet(int fleetId)
על הקלט $O(1)$ מחזירה את מספר המטבעות של $O(1)$ מחזירה את מספר המטבעות של $O(log^*m)$ בממוצע על הקלט $O(log^*m)$	\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \		
$0(1)$ פיראט בעל מזהה מספר המטבעות של פיראט בעל מזהה pirateld פיראט בעל מזהה pirateld איחוד הציים בעלי המזהים fleetId1 איחוד הציים בעלי המזהים fleetId1, int fleetId2) איחוד הציים בעלי המזהה של הם משוערך בממוצע משוערך בממוצע הוא המזהה של הצי הגדול יותר על הקלט של שוויון המזהה החדש הוא מווחד הוא סכום מספר הספינות בצי של שוויון המזהה החדש הוא מכום מספר הספינות בצי המאוחד הוא סכום מספר הספינות בצי הפיראטים בצי הקטן יותר עגדל בשני הציים, והדרגה של כל הפיראטים בצי הקטן יותר עגדל למר הפיראטים בצי הקטן יותר צי עם מספר הפיראטים בצי הקטן, ולא ניתן לאחר האיחוד לא קיים יותר צי עם להוסיף צי חדש עם מזהה זה. $0(log^*m)$ וויכוח בין שני פיראטים. הפיראט משוערך בממוצע משוערך בממוצע של הקלט משוערך בממוצע של הפיראט של הפיראט על הקלט שדרגתו נמוכה יותר ישלם לפיראט על הקלט על הקלט שדרגתו נמוכה יותר יסכום הכסף על הקלט		,	
פיראט בעל מזהה $pirateld$ בממוצע על הקלט $O(log^*m)$ איחוד הציים בעלי המזהים $fleetId2$ ו- $fleetId2$ המזהה החדש שלהם משוערך בממוצע $graphicald2$ המזהה של הצי הגדול יותר על הקלט $graphicald2$ בממינות בצי של שוויון המזהה החדש הוא $graphicald2$ בשני הציים, והדרגה של כל המאוחד הוא סכום מספר הפימות $graphicald2$ בשני הציים, והדרגה של כל בשני הציים, והדרגה של כל במספר הפיראטים בצי הגדול יותר. הפיראטים בצי הגדול יותר. המזהה של הצי עם במספר הפיראטים בצי הגדול יותר. המזהה של הצי הקטן, ולא ניתן לאחר האיחוד לא קיים יותר צי עם להוסיף צי חדש עם מזהה זה. $graphicald2$ $graphicald3$		מחזירה את מספר המטבעות של	output t < int > get pirate monev(int pirateId)
$O(log^*m)$ איחוד הציים בעלי המזהים לובפלו איחוד הציים בעלי המזהה לובפלו איחוד הציים בעלי המזהה שלהם הוא החדש שלהם הוא המזהה של הצי הגדול יותר על הקלט של שוויון המזהה החדש הוא המאוחד הוא סכום מספר הספינות בצי המאוחד הוא סכום מספר הספינות בשי הנדול יותר. הפיראטים בצי הקטן יותר תגדל בשני הציים, והדרגה של כל מספר הפיראטים בצי הקטן יותר תגדל לאחר האיחוד לא קיים יותר צי עם למחר האיחוד לא קיים יותר צי עם להוסיף צי חדש עם מזהה זה. **StatusType unite_fleets(int fleetId1, int fleetId2)** **Veraginal Author of the properties of th			3.4-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
וויכוח בין שני פיראטים. משוערך בממוצע משוערך בממוצע אל הקלט מבחינת מספר הפיראטים (במקרה של של שוויון המזהה החדש הוא (fleetId1 בשני הציים, והדרגה של כל בשני הציים, והדרגה של כל בשני הציים, והדרגה של כל הפיראטים בצי הקטן יותר תגדל במספר הפיראטים בצי הגדול יותר. המזהה של הצי הקטן, ולא ניתן לאחר האיחוד לא קיים יותר צי עם המזהה של הצי הקטן, ולא ניתן להוסיף צי חדש עם מזהה זה. StatusType pirate_argument(int pirateId1, int pirateId2) שדרגתו גבוהה יותר ישלם לפיראט של הקלט שדרגתו נמוכה יותר. סכום הכסף על הקלט			StatusType unite fleets(int fleetId1, int fleetId2)
הוא המזהה של הצי הגדול יותר מבחינת מספר הפיראטים (במקרה של שוויון המזהה החדש הוא של שוויון המזהה החדש הוא המאוחד הוא סכום מספר הספינות בצי בשני הציים, והדרגה של כל בשני הציים, והדרגה של כל הפיראטים בצי הקטן יותר תגדל כמספר הפיראטים בצי הגדול יותר. לאחר האיחוד לא קיים יותר צי עם לאחר האיחוד לא קיים יותר צי עם המזהה של הצי הקטן, ולא ניתן להוסיף צי חדש עם מזהה זה. StatusType pirate_argument(int pirateId1, int pirateId2) שדרגתו גבוהה יותר ישלם לפיראט שדרגתו נמוכה יותר. סכום הכסף על הקלט			
מבחינת מספר הפיראטים (במקרה של שוויון המזהה החדש הוא של שוויון המזהה החדש הוא (fleetId1). מספר הספינות בצי (fleetId1). מספר הספינות בצי בשני הציים, והדרגה של כל בשני הציים, והדרגה של כל בשני הציים, והדרגה של כל במספר הפיראטים בצי הקטן יותר תגדל כמספר הפיראטים בצי הגדול יותר. לאחר האיחוד לא קיים יותר צי עם לאחר האיחוד לא קיים יותר צי עם המזהה של הצי הקטן, ולא ניתן להוסיף צי חדש עם מזהה זה. בוהה יותר שלם לפיראט משוערך בממוצע שדרגתו גבוהה יותר ישלם לפיראט על הקלט שדרגתו נמוכה יותר. סכום הכסף על הקלט שדרגתו נמוכה יותר. סכום הכסף			
של שוויון המזהה החדש הוא $(fleetId1)$ מספר הספינות בצי ($fleetId1$). מספר הספינות בצי המאוחד הוא סכום מספר הספינות בשני הציים, והדרגה של כל הפיראטים בצי הקטן יותר תגדל כמספר הפיראטים בצי הגדול יותר. לאחר האיחוד לא קיים יותר צי עם לאחר האיחוד לא קיים יותר צי עם המזהה של הצי הקטן, ולא ניתן להוסיף צי חדש עם מזהה זה. $O(log^*m)$ וויכוח בין שני פיראטים. הפיראט $O(log^*m)$ StatusType pirate_argument(int pirateId1, int pirateId2) שדרגתו גבוהה יותר ישלם לפיראט על הקלט שדרגתו נמוכה יותר. סכום הכסף על הקלט	·	מבחינת מספר הפיראטים (במקרה	
א מספר הספינות בצי המאוחד הוא סכום מספר הספינות המאוחד הוא סכום מספר הספינות המאוחד הוא סכום מספר הספינות בשני הציים, והדרגה של כל הפיראטים בצי הקטן יותר תגדל כמספר הפיראטים בצי הגדול יותר. לאחר האיחוד לא קיים יותר צי עם לאחר האיחוד לא קיים יותר צי עם המזהה של הצי הקטן, ולא ניתן להוסיף צי חדש עם מזהה זה. $O(log^*m)$ וויכוח בין שני פיראטים. הפיראט המשוערך בממוצע StatusType pirate_argument(int pirateId1, int pirateId2) שדרגתו גבוהה יותר ישלם לפיראט על הקלט שדרגתו נמוכה יותר. סכום הכסף על הקלט			
המאוחד הוא סכום מספר הספינות המאוחד הוא סכום מספר הספינות בשני הציים, והדרגה של כל הפיראטים בצי הקטן יותר תגדל כמספר הפיראטים בצי הגדול יותר. ממספר הפיראטים בצי הגדול יותר. לאחר האיחוד לא קיים יותר צי עם המזהה של הצי הקטן, ולא ניתן להוסיף צי חדש עם מזהה זה. הפיראט משוערך בממוצע $O(log^*m)$ שדרגתו בוהה יותר ישלם לפיראט משוערך בממוצע שדרגתו גבוהה יותר ישלם לפיראט על הקלט על הקלט			
הפיראטים בצי הקטן יותר תגדל כמספר הפיראטים בצי הגדול יותר. לאחר האיחוד לא קיים יותר צי עם לאחר האיחוד לא קיים יותר צי עם המזהה של הצי הקטן, ולא ניתן להוסיף צי חדש עם מזהה זה. O(log*m) וויכוח בין שני פיראט StatusType pirate_argument(int pirateId1, int pirateId2) שדרגתו גבוהה יותר ישלם לפיראט על הקלט שדרגתו נמוכה יותר. סכום הכסף על הקלט			
הפיראטים בצי הקטן יותר תגדל כמספר הפיראטים בצי הגדול יותר. לאחר האיחוד לא קיים יותר צי עם לאחר האיחוד לא קיים יותר צי עם המזהה של הצי הקטן, ולא ניתן להוסיף צי חדש עם מזהה זה. **StatusType pirate_argument(int pirateId1, int pirateId2)** שדרגתו גבוהה יותר ישלם לפיראט על הקלט שדרגתו נמוכה יותר. סכום הכסף על הקלט			
כמספר הפיראטים בי הגדול יותר. לאחר האיחוד לא קיים יותר צי עם לאחר האיחוד לא קיים יותר צי עם המזהה של הצי הקטן, ולא ניתן להוסיף צי חדש עם מזהה זה. $O(log^* m))$ וויכוח בין שני פיראטים. הפיראט הפיראט StatusType pirate_argument(int pirateId1, int pirateId2) שדרגתו גבוהה יותר ישלם לפיראט על הקלט שדרגתו נמוכה יותר. סכום הכסף על הקלט			
לאחר האיחוד לא קיים יותר צי עם המזהה של הצי הקטן, ולא ניתן המזהה של הצי הקטן, ולא ניתן להוסיף צי חדש עם מזהה זה. להוסיף צי חדש עם מזהה זה. $ O(log^*m)) $ וויכוח בין שני פיראטים. הפיראט שדרגתו בוהה יותר ישלם לפיראט שוערך בממוצע שדרגתו גבוהה יותר ישלם לפיראט שדרגתו נמוכה יותר. סכום הכסף על הקלט			
המזהה של הצי הקטן, ולא ניתן המזהה של הצי הקטן, ולא ניתן המזהה של הצי הקטן, ולא ניתן להוסיף צי חדש עם מזהה זה. $O(log^*m))$ אויכוח בין שני פיראטים. הפיראט משוערך בממוצע שדרגתו גבוהה יותר ישלם לפיראט משוערך בממוצע שדרגתו נמוכה יותר. סכום הכסף על הקלט			
להוסיף צי חדש עם מזהה זה. להוסיף צי חדש עם מזהה זה. אלהוסיף צי חדש עם מזהה זה. להוסיף צי חדש עם מזהה זה. $ StatusType\ pirate_argument(int\ pirateId1,int\ pirateId2) $ שדרגתו גבוהה יותר ישלם לפיראט שוערך בממוצע שדרגתו נמוכה יותר. סכום הכסף על הקלט			
אינור פון שני פיראטים. הפיראט (Interpretable $O(log^*m)$) אויכוח בין שני פיראטים. הפיראט (StatusType pirate_argument(int pirateId1, int pirateId2) שדרגתו גבוהה יותר ישלם לפיראט (שדרגתו נמוכה יותר. סכום הכסף על הקלט			
שדרגתו גבוהה יותר ישלם לפיראט משוערך בממוצע שדרגתו נמוכה יותר. סכום הכסף על הקלט על הקלט		'	
שדרגתו גבוהה יותר ישלם לפיראט משוערך בממוצע שדרגתו נמוכה יותר. סכום הכסף על הקלט על הקלט	$O(log^* m))$	וויכוח בין שני פיראטים. הפיראט	StatusType pirate_argument(int pirateId1, int pirateId2)
שדרגתו נמוכה יותר. סכום הכסף על הקלט	משוערך בממוצע		- · · ·
	-	שדרגתו נמוכה יותר. סכום הכסף	
		נקבע לפי הפרש הדרגות.	

כאשר n הוא מספר הפיראטים ו-m הוא מספר הציים שהתווספו למערכת לאורך ריצת התכנית.

. במבנה הנתונים נדרש לעמוד בהגבלה של סיבוכיות זיכרון של סיבונה עצמו, ובכל הפעולות. בנוסף, ממבנה הנתונים נדרש לעמוד בהגבלה של סיבוכיות דיכרון

<u>מימוש:</u>

מימשנו את מבנה הנתונים באמצעות הרכבה של מבני נתונים קיימים וידועים, בעלי סיבוכיות זמן ריצה ומקום מימשנו את מבנה הנתונים באמצעות הרכבה של מבני נתונים קיימים וידועים (UpTree).

במבנה הנתונים שלנו, יש סימון מבלבל (בעץ ההפוך) בין דרישות מבנה הנתונים לאופן המימוש שלנו, ולכן נחלק את הסבר הפתרון לשני חלקים:

- הסבר על ההרכבה של מבנה הנתונים הגנרי, של טבלאות ערבול ועץ הפוך
 - הסבר על ההתאמה של מבנה הנתונים הגנרי לבעיה שהוצגה בשאלה

אופן ההרכבה של מבני הנתונים הגנרי:

הגדרות:

- לאיברים בעץ ההפוך נקרא <u>קבוצות</u>.
- לאחר איחוד קבוצות, נאמר כי הקבוצה הקטנה יותר <u>אוחדה מלמטה</u> (לאחר האיחוד, קבוצה זו מחזיקה במצביע לקבוצה האחרת, ונאמר שהקבוצה האחרת אוחדה מלמעלה).
- לאחר איחוד קבוצות, נאמר כי הקבוצה שאוחדה מלמטה מחזיקה מצביע לקבוצה <u>מעל</u> (הקבוצה שאוחדה מלמעלה).
 - קבוצה שעד כה לא אוחדה מלמטה תקרא <u>קבוצת על</u>.
 - לאיברים שמשתייכים לקבוצות, נקרא <u>תאים</u>.
 - כאשר מוסיפים תא לקבוצה, נאמר כי התא <u>משוייך</u> לקבוצה זו במבנה הנתונים.

הערות:

השדות של קבוצה הם:

- מזהה (לא בהכרח מספר, אך יש לו מספר כאחד השדות)
 - מצביע (ריק אם זו קבוצת על) לקבוצה מעל •
- גודל הקבוצה (מספר הקבוצות הכולל תחתיה, כולל היא עצמה)
 - (בדומה לעץ בינארי רגיל) •

השדות של תא הם:

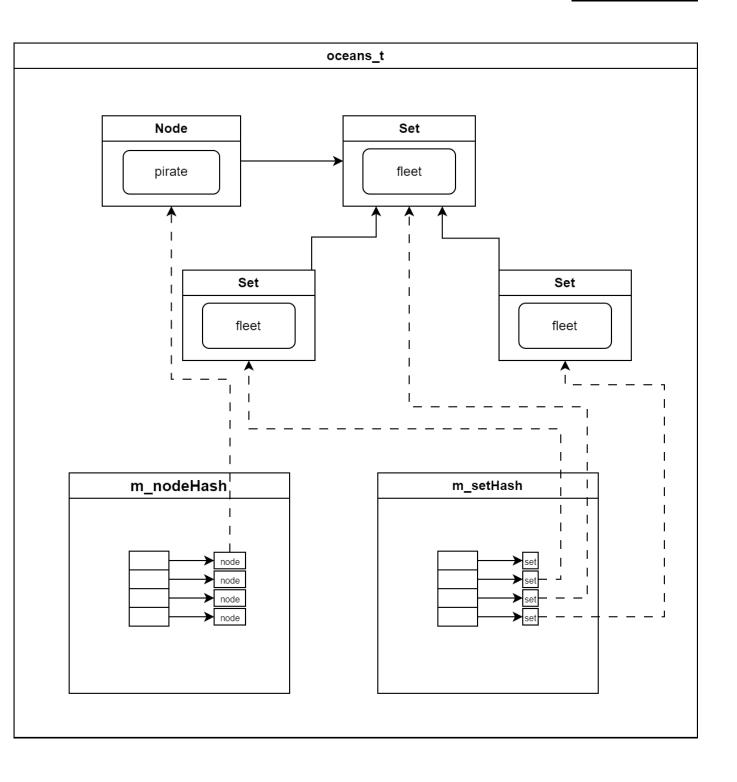
- מזהה (לא בהכרח מספר, אבל יש לו מספר כאחד השדות)
 - מצביע לקבוצה אליה הוא משוייך •

<u>אבחנה:</u> העץ ההפוך הוא בעצם עץ הפוך של קבוצות.

במבנה הנתונים שלנו 2 טבלאות ערבול:

- <u>טבלת הקבוצות</u>- טבלת ערבול, אשר ממפה (מלשון מיפוי) מזהה של קבוצה למצביע לאותה קבוצה בעץ ההפוך.
 - <u>טבלת התאים</u>- טבלת ערבול, אשר ממפה (מלשון מיפוי) מזהה של התא למצביע לאותו התא.

"<u>רעיונית-</u> נרצה ש"במקום קבוצות יהיו ציים, ובמקום תאים יהיו פיראטים



<u>סיבוכיות הזיכרון של מבנה הנתונים:</u>

. בהרצאות ראינו כי סיבוכיות הזיכרון הן HashTable והן של עץ הפוך היא

נניח כעת כי סיבוכיות הזיכרון של תא וגם של קבוצה היא O(1) (נדרוש זאת בהמשך במעבר לפיראטים וציים) ונחסום את סיבוכיות הזיכרון במבנה הנתונים:

נסמן ב-m את מספר הקבוצות, וב-n את מספר התאים.

- O(m) -טבלת ערבול של קבוצות
 - .O(n) -טבלת ערבול של תאים •
- עץ הפוך של קבוצות- O(m). (העץ אמנם מבוסס על הזיכרון של טבלת ערבול הקבוצות אז אפשר לומר שאין לו תרומה לסיבוכיות הזיכרון)

 $O(m) + O(n) \left(+ O(m) \right) = O(n+m)$ לכן, סה"כ סיבוכיות הזיכרון של מבנה הנתונים היא

לכן, כאשר נתאים קבוצות \to ציים, תאים \to פיראטים, כך שתתקיים הדרישה לעיל (סיבוכיות זיכרון (0(1))), נקבל סיבוכיות זיכרון (n+m) כאשר m מספר הציים אשר נוספו למבנה הנתונים במהלך הריצה, n מספר הפיראטים, כנדרש.

נוכיח את סיבוכיות הזמן של הפעולות הבאות על מבנה הנתונים הגנרי:

0(1)	אתחול מבנה נתונים ריק. קבוצות ותאים.	init()
במקרה הגרוע		
O(n+m)	שחרור מבנה הנתונים וכל הזיכרון.	~free()
במקרה הגרוע		
0(1)	setId יצירת קבוצה חדשה בעל מזהה	StatusType makeSet(const S& setId)
משוערך בממוצע על	(שאינו בהכרח מספר)	
הקלט		
$O(log^* m)$	בהינתן מזהה של קבוצה (מספר), יש למצוא	output_t < Set *> findSet(int setId)
משוערך בממוצע על	את המצביע <u>לקבוצת העל</u> של הקבוצה בעל	
הקלט	המזהה	
$O(log^* m)$	בהינתן מזהים של שתי קבוצות (מספרים),	StatusType union2Sets(int set1Id, int set2Id)
משוערך בממוצע על	לאחד את <u>קבוצות העל</u> של הקבוצות בעלי	
הקלט	אותם מזהים.	
$O(log^* m)$	שיוך תא ל <u>קבוצת העל</u> של הקבוצה בעל	StatusType insertValue(int setId, const T& value)
משוערך בממוצע על	המזהה (המספרי) setId.	
הקלט		
0(1)	בהינתן מזהה (מספרי) של תא, הפעולה	output_t < Set *> fetchSetOf(int valueId)
בממוצע על הקלט	מחזירה מצביע לקבוצה אליה התא משוייך.	
0(1)	בהינתן מזהה (מספרי) של תא, הפעולה	output_t < T *> fetch(int valueId)
בממוצע על הקלט	מחזירה את המצביע למזהה (לא מספרי) של	
	אותו התא.	

. אתחול טבלאות ערבול ריקות, סיבוכיות זמן (0(1) במקרה הגרוע :init()

:∼free() <u>הפעולה</u>

שחרור הזיכרון של טבלת הערבול של התאים- O(n) במקרה הגרוע, ושחרור הזיכרון של טבלת הערבול של התאים- O(m).

לכן, סה"כ סיבוכיות זמן O(n+m) במקרה הגרוע.

:makeSet(const S& setId) הפעולה

- . יוצרים קבוצה חדשה (קבוצת על) בעלת המזהה setId, סיבוכיות זמן O(1) במקרה הגרוע.
- מכניסים לטבלת הערבול את המזהה (המזהה המספרי של setId) עם המצביע לקבוצה שנוצרה.

.0(1) כיוון שזו טבלת ערבול, מובטח לנו סיבוכיות זמן ריצה משוערכת בממוצע על הקלט

:findSet(int setId) הפעולה

- שולפים מתוך טבלת הערבול של הקבוצות את הקבוצה בעלת המזהה (המספרי) setId.
 - מוצאים את המצביע של קבוצת העל באופן הבא: (נאתחל את סכום הדרגות ל-0)
 - בודקים האם המצביע של הקבוצה מעל הוא ריק-
 - אם כן, מחזירים את המצביע של הקבוצה הנוכחית
 - אחרת, о
 - נוסיף לסכום הדרגות את הדרגה של הקבוצה הנוכחית.
 - עוברים לקבוצה מעל ומפעילים עליה את אותו אלגוריתם.
 - :כיווץ הענף בעץ
 - ניקח את המצביע לקבוצה ששלפנו מטבלת הערבול, ונפעל כך: 🌼
 - י נשנה את המצביע לקבוצה מעל כך שיצביע לקבוצת העל.
 - . נחסיר מסכום הדרגות את הדרגה של הקבוצה הנוכחית.
- נעדכן את הדרגה של הקבוצה הנוכחית להיות הדרגה + סכום הדרגות.
- . נפעיל את האלגוריתם על הקבוצה (שהייתה) מעל אלא אם הגענו לשורש.
 - מחזירים את המצביע של קבוצת העל.

.0(1) גישה לאיבר מתוך טבלת ערבול זו פעולה בסיבוכיות זמן ריצה משוערכת בממוצע על הקלט

שני האלגוריתמים, של מציאת המצביע לקבוצת העל, ושל כיווץ הענף בעץ הם פעולות רקורסיביות, כאשר עומק הרקורסיה חסום ע"י גובה העץ ההפוך, וחישובי הביניים הם O(1) במקרה הגרוע.

לכן (מובטח חסם משוערך בממוצע על הקלט $O(\log^* m)$ לגובה העץ ההפוך כאשר m זהו מספר הקבוצות שהוכנסו למבני הנתונים לאורך הריצה), סיבוכיות הזמן של פעולות אלו היא $O(\log^* m)$ משוערך בממוצע על הקלט.

סה"כ, סיבוכיות זמן הריצה של הפעולה היא $O(\log^* m) + O(1) = O(\log^* m)$ משוערך בממוצע על הקלט.

:union2Sets(int set1Id, int set2Id) הפעולה

- . findSet מוצאים את קבוצות העל של הקבוצות בעלי המזהים (המספריים) הנתונים, באמצעות -
 - O(1), מבצעים איחוד על שתי קבוצות העל (הקבוצה הקטנה יותר מביניהן תאוחד מלמטה),
- $oldsymbol{0}$ מחסירים מדרגת הקבוצה שאוחדה מלמטה את דרגת הקבוצה שאוחדה מלמעלה, סיבוכיות (O(1)
 - .(unite_fleets נבצע עדכון דרגות לפי המזהים (כפי שיותאר בפעולה •

. הפעולה היא בסיבוכיות זמן משוערכת בממוצע על הקלט $O(\log^* m)$, ולכן גם סיבוכיות הפעולה הזו.

:insertValue(int setId, const T& value) הפעולה

- .findSet באמצעות, setId מוצאים את קבוצת העל של הקבוצה בעלת המזהה
 - .0(1) סיבוכיות, value משייכים לקבוצת העל את התא עם המזהה \bullet

 $O(\log^* m)$ סה"כ, סיבוכיות זמן ריצה מושערכת בממוצע על הקלט

:fetchSetOf(int valueId) הפעולה

- שולפים את המצביע לתא בעל המזהה (המספרי) valueId מתוך טבלת הערבול של התאים. •
- O(1) מחזירים את המצביע של הקבוצה אליה משוייך התא (אחד מהשדות של התא), סיבוכיות O(1)

בטבלת ערבול מובטחת סיבוכיות זמן ממוצעת על הקלט O(1) על פעולת שליפת ערך לפי מפתח. לכן סיבוכיות הזמן של פעולה זו היא O(1) בממוצע על הקלט.

:fetch(int valueId) הפעולה

- שולפים את המצביע לתא בעל המזהה (המספרי) valueId מתוך טבלת הערבול של התאים.
 - 0(1) מחזירים את המצביע של מזהה התא (אחד השדות של התא), סיבוכיות

בטבלת ערבול מובטחת סיבוכיות זמן ממוצעת על הקלט O(1) על פעולת שליפת ערך לפי מפתח. לכן סיבוכיות הזמן של פעולה זו היא O(1) בממוצע על הקלט.

הערות על הוכחת חישוב סיבוכיות הזמנים המשוערכים בממוצע על הקלט:

- הפעולה findSet מלבד הנגיעה בטבלת הערבול של הקבוצות, ועדכון הדרגות (שאינו משפיע על הפעולה לסיבוכיות זמן הריצה), זהה לפעולה find של עץ הפוך כפי שנלמדה בהרצאה, אשר מובילה לסיבוכיות זמן הריצה), זמן $O(\log^* m)$ ים שבעץ ההפוך).
- לכן, כל אחת מהפעולות של מבנה הנתונים (מלבד שחרורו) מפעילה עד 2 פעולות על כל אחד ממבני הנתונים, טבלאות הערבול, והעץ ההפוך. לכן, סיבוכיות הזמן המשוערכת נשמרת.

התאמת מבנה הנתונים הגנרי לבעיה שהוצגה בשאלה

מבנה הנתונים הגנרי כמעט מהווה פתרון בעצמו לדרישות מבנה הנתונים. אבל יש בעיה שחייבים לפתור:

הקונפליקט בהתאמת המבנה לציים:

כאשר מאחדים שני ציים (לפי מבנה הנתונים הגנרי כפי שהוצג), הצי בעל מספר הספינות הגבוה יותר מאוחד מלמעלה (נבחין כי גודל הקבוצה הוא מספר הספינות של הציים בקבוצות העל), והוא הופך לקבוצת העל.

-מצד שני

מדרישות מבנה הנתונים, מזהה קבוצת העל "צריך" להיות של הצי עם מספר ה**פיראטים** הגדול יותר.

כיוון שאין קשר בין מספר הספינות תחת צי לבין מספר הפיראטים תחתיו, עלולים להיות מקרים בהם הסדר לפי מבנה הנתונים הגנרי הוא "הפוך" ממה שצריך להיות בדרישה ממבנה הנתונים.

אתגר נוסף (שהופך לקשה יותר בגלל הקונפליקט) הוא מתן דרגה לפיראט ועידכונה במהלך שינויים במבנה הנתונים (בלי פעולות מיותרות שמשפיעות על סיבוכיות הזמן).

הפתרון הוא כולו באופן מימוש המזהה של הקבוצה, כלומר אובייקט הצי (מימוש אובייקט הפיראט טריוויאלי). תזכורת- פיראט יהיה מזהה של תא, וצי יהיה מזהה של קבוצה.

שדות הפיראט:

- מזהה (מספרי)
- אוצר (כמות מטבעות) •
- דרגתו בצי אליו השתייך (דרגת בסיס)

שדות הצי:

- מזהה (מספרי)
- מספר ספינות תחת הצי
- מספר פיראטים תחת הצי
- מאזן הדרגה של הצי (הדרגה שצריך להוסיף לכל מי שנמצא תחת אותו צי), זו הדרגה של הקבוצה.
- מזהה דורס- מזהה הספינה שבפועל היא בעלת מספר הפיראטים הגבוה יותר (תוצאת האיחוד לפי הדרישות)

נותר רק לתאר את הפעולות ולהשתמש בחסמי סיבוכיות הזמן של מבנה הנתונים הגנרי שהוכחנו.

האובייקטים של הפיראט והצי עומדים בדרישת הסיבוכיות זיכרון 0(1). ולכן הפעולות של יצירת מבנה נתונים coceans_t, ריק ו-מחיקת מבנה הנתונים ושחרורו עומדים בסיבוכיות הזמן 0(1) ליצירת מבנה הנתונים, oceans_t ריק ו-מחיקת מבנה הנתונים, 0(n+m) לפעולה coceans_t.

(מספר הציים = מספר הקבוצות, מספר הפיראטים = מספר התאים)

:add_fleet(int fleetId) הפעולה

ניצור אובייקט צי עם המזהה fleetId, ספינה אחת תחתיו, 0 פיראטים תחתיו, מאזן דרגה 0.

נכניס את האובייקט הזה למבנה הנתונים הגנרי באמצעות makeSet.

סיבוכיות זמן הריצה היא O(1) משוערך בממוצע על הקלט, לפי הדרישות.

<u>:add_pirate(int pirateId, int fleetId) הפעולה</u>

- .0(1) עם 0 מטבעות. pirateId ניצור אובייקט פיראט עם המזהה
- נשייך את הפיראט שייצרנו לצי של קבוצת העל של הקבוצה עבורה מזהה הצי הוא fleetId, באמצעות הפעולה insertValue.
- 1+, נגדיר את דרגת הבסיס של הפיראט להיות מספר הפיראטים בצי של קבוצת העל לפני הוספתו גדיר את דרגה הבסיס של הפיראט להיות מספר הפיראט יוסיפו את מאזן הדרגה של הצי (כך כאשר ישאלו על דרגת הפיראט יוסיפו את מאזן הדרגה של הצי x+1, כמו בדרישות מבנה הנתונים.) (0(1).

לכן, סה"כ סיבוכיות זמן הריצה הוא $O(\log^* m)$ משוערך הממוצע על הקלט, לפי הדרישות.

<u>:pay pirate(int pirateId, int salary) הפעולה</u>

שולפים את הפיראט בעל המזהה (המספרי) pirateId מתוך מבנה הנתונים, באמצעות הפונקציה fetch.

.salary- משנים את כמות המטבעות של הפיראט

סיבוכיות זמן הריצה היא O(1) בממוצע על הקלט, לפי הדרישות.

:num ships for fleet(int fleetId) הפונקציה

מוצאים את מצביע הצי של קבוצת העל של הקבוצה עבורה מזהה הצי הוא fleetId, באמצעות findSet

.0(1) מחזירים את מספר הספינות תחת צי זה (שזה גם מספר הקבוצות תחת קבוצת העל), מחזירים

לכן, סה"כ סיבוכיות זמן הריצה היא $O(\log^* m)$ משוערך בממוצע על הקלט, לפי הדרישות.

:get pirate money(int pirateId) הפעולה

שולפים את הפיראט בעל המזהה (המספרי) pirateId מתוך מבנה הנתונים, באמצעות הפונקציה

O(1) מחזירים את כמות המטבעות של הפיראט,

סיבוכיות זמן הריצה היא O(1) בממוצע על הקלט, לפי הדרישות.

<u>:unite fleets(int fleetId1, int fleetId2)</u>

באמצעות הפעולה union2Sets, נאחד את קבוצות העל של הקבוצות עבורן הציים בעלי המזהים הנתונים.

כדי לאפשר נכונות מבחינת דרישות מבנה הנתונים, נרצה לזכור את מזהה הצי שאמור להיות הצי בקבוצת העל מבחינת מבנה הנתונים, בקבוצת העל. לשם כך, לאחר איחוד לפי מספר הספינות (גדלי הקבוצות), לקבוצה שאוחדה מעל נשנה את המזהה הדורס של הצי שלה להיות המזהה הדורס של הצי של הקבוצה שאוחדה מלמטה יש יותר פיראטים, או שיש בדיוק אותו מספר פיראטים, אבל הסדר לפי דרישות מבני הנתונים הוא הפוך.

בנוסף, צריך לעדכן את דרגות כל הפיראטים בצי שבו יש פחות פיראטים לגדול במספר הפיראטים בצי האחר. כמו בעדכון המזהה הדורס, נחלק למקרים:

אם הסדר נכון (הצי שאוחד מלמעלה אמור להיות מאוחד מלמעלה לפי דרישות מבנה הנתונים):

• נגדיל את מאזן הדרגה של הצי שאוחד מלמטה במספר הפיראטים של הצי שאוחד מלמעלה.

אחרת (הסדר הפוך, כלומר לכל הפיראטים בצי שאוחד מלמעלה ולא בצי שאוחד מלמטה צריך להגדיל דרגה),

- נגדיל את מאזן הדרגה של הצי שאוחד מלמעלה במספר הפיראטים של הצי שאוחד מלטה.
- כדי להחזיר את הדרגה הכוללת של הפיראטים שהיו בצי שאוחד מלמטה או תחתיו, נקטין את מאזן הדרגה של הצי שאוחד מלמטה במספר הפיראטים של הצי שאוחד מלטה.

לבסוף, נעדכן את מספר הספינות בצי שאוחד מלמעלה לסכום הספינות בשני הציים (של קבוצות העל), וכך גם נעדכן את מספר הפיראטים (להיות הסכום).

לכן, סה"כ סיבוכיות זמן הריצה היא $O(\log^* m)$ משוערך בממוצע על הקלט, לפי הדרישות.

:pirate argument(int pirateId1, int pirateId2) הפעולה

מוצאים את הפיראטים וקבוצות העל של הציים שלהם באמצעות הפונקציות fetch ,findSet.

לפי תוצאת findSet, ניתן בקלות לבדוק האם הפיראטים תחת אותו צי- האם קבוצת העל שלהן משותפת, כלומר אותה תוצאת חיפוש. בנוסף, פעולה זו קיצצה את עומק העץ החל מהציים המקוריים של הפיראטים עד לקבוצת העל לאורך $\mathcal{O}(1)$.

.O(1)-ב ניתן לחשב את דרגת הפיראטים, fetchSetOf לכן, באמצעות

(דרגת הבסיס, כתוצאה מ-fetch + סכימת דרגות המאזן של הציים בדרך, כאשר יש עד צי אחד מעל).

O(1) מציאת הפרשי דרגות הפיראטים, ושינוי שדות כמות המטבעות של הפיראטים בהתאם, לבסוף, מציאת הפרשי דרגות הפיראטים, ושינוי שדות כמות המוצע על הקלט. לכן, סה"כ סיבוכיות הזמן היא $O(\log^* m)$ משוערך בממוצע על הקלט, לפי הדרישות.