

Union-Find to struktura danych przydatna przy optymalizacji implementacji algorytmu Kruskala.

### Algorytm Kruskala

Mając zbiór wierzchołków V i krawędzi E grafu  $G = (V, E)$  szukamy minimalnego okręgu rozpinającego  $G$  - MST.

Każdy wierzchołek grafu jest traktowany jako osobne podzialewo. Przechodząc przez krawędzie ułożone w rosnącej kolejności wag. jeśli dwa krańce krawędzi są należące do różnych podzialew, taka krawędź jest ujęta w podzialewo. Postępując operacją, aż wszystkie krawędzie będą w jednym podzialewo - MST.

### Idea Union-Find

Tużymy strukturę, która wspomaga operacje sprawdzania, do którego podzialew należy wierzchołek  $v$  ( $\text{Find}(v)$ ) orazłączenia podzialew  $x, y$  ( $\text{Union}(x, y)$ ).

### I wersja (najprostsza)

Najprostsza implementacja polega na utrzymaniu tablicy  $R[1..n]$ , gdzie  $R[i]$  to numer podzialewu elementu  $i$ .  $\text{Find}(i)$  kosztuje  $\Theta(1)$ , łączystwo  $\Theta(n)$ .

$\text{Union}(x, y)$  może kosztować  $\Theta(n)$ , gdy zbiory będą odpowiadających im podzialewów zmieniają swoistosć uszeregowania w  $R$  dla elementów z  $x, y$ . Ciąg operacji  $\text{Union}$  może mieć kosztową  $\Theta(n^2)$ .

### II wersja (usprawniona)

Zauważmy, że mamy:

1) nie interesują nas nowe podzialewy - chcemy tylko połączić elementy należące do jednego lub dwóch różnych obiektów. Np. wykonyując  $\text{Union}(x, y)$  mamy na pewno połączonych elementów  $x, y$  ale nowego zbiornika, a nie połączonych elementów nowego z  $x, y$  ale połączonych.

2) dany podzialewy  $A, B$  możemy połączić mniej więcej - wykonywać takie same operacje.

Elementy jednego zbiornika przedstawiający na liście ułączane. Główki jest reprezentantem zbiornika. Każdy element zbiornika ma wskaznik na tego reprezentanta.

Find(i) - koszt  $\Theta(1)$ , ponieważ jest to zapis poł. wskaznika  $i \rightarrow$  head.

Union(A,B) - koszt  $\Theta(\min(|A|, |B|))$ , ponieważ mamy przepis elementy krotszy losty o odkinie - mamy wtedy pamiętnie zmienią lost.

Złożoność cięgu operacji Union-Find to  $\Theta(m + n \log n)$ .

dl-dl

Algorytm konczy się w  $2m$  operacjach Find, gdzie  $m$  to suma liczb elementów (spowodzona głębokością drzewa lub liczba drzew). Dla naszej złożoności cięgu tych operacji to  $\Theta(m) \cdot \Theta(1) = \Theta(m)$ .

Operacja Union wykonyje serię przepisów skrótów, które zajmuje  $\Theta(1)$  czasu. Ile razy konieczne jest zmiana przypisanej elementu  $x$ ?

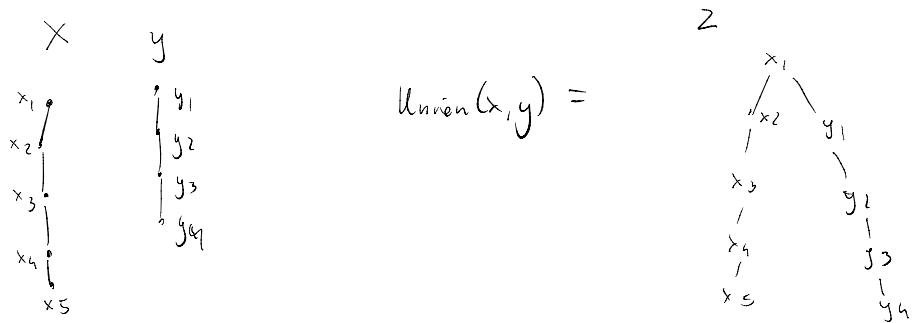
Zauważ, że nie powtarza się żaden z elementów i daje nam go o którym dotyczą losty  $y$ . Wtedy czasu  $y$  musi być więcej niż  $|x|$ , aby  $|x \cup y| = |x| + |y| > 2$ , czyli  $x$  należy do co najmniej dwóch różnych lostów. Oznacza to, że  $x$  może przypisać losty  $B$ , takie że  $|B| > |A|$ , zatem  $|C| = |B \cup A| > 2|A|$ . Oznacza to, że  $x$  może przypisać co najmniej  $\log n$  różne, ponieważ  $2^{\log n} = n$ , aby po tym przepisach zmienić się w listę o długości  $n$ , aby było jedynie jedna lista. Mamy  $n$  wartości, zatem czas wykonywania  $\Theta(n \log n)$  przypisów, co daje złożoność cięgu Unionów.

Dla naszej cięgi Unionów i Findów kosztuje  $\Theta(m + n \log n)$ .

### III wersja (szybsza)

Zauważmy, że taki napisany przekształcanie zbiorów w postaci lost jest zbyt tecznym - jedyne mamy potrzebujemy, to abyśmy mogli reprezentować zbioru. W takim razie mamy uproszczyć operacje Union aby przepisów dotyczących mnóstw lostów przestały być potrzebne.

Union ma przepisów dotyczących mnóstw lostów przestały być potrzebne.



l i t . . . 1 + 1 -> ... l . . . oznacza, że jeśli wartości mamy oznaczyć losty, to

U ten sposób fazyne daje spełnione obowiązków, jednak warunków może mniej niż w kierunku sygnału, ale nie muszą być tym samymi, bo kierunek warunków pozwala jedynie na określonej mocy. Mocą powinno kierować finalną fazą fazy, ale zależy od wyników technicznych danej (mocą jest całkowita wartość wartości w doborze danej - reprezentantów). Dostępne są dwa typy decyzyjne i strategie przyjmowania rozwiązań, o których mowa będzie już teraz skróconym określeniem, ale będą braków szczegółów (przykład).

Wprowadzony obraz kierunków:

1) przyjmowanie o mocy, której reguluje połkrotnie kierunek tego oznaczanego rozwiązań. Zadanie zadającym regul. (zbalsansowany Union)

2) podawanie kierunku finalnego przyjętego w fazie warunków bez określania połkrotności kierunku danej. Utyczne kiedy mamy określone problemy techniczne - kierunki kierunki i dany, lecz przyjęte zobowiązania. (kompreksja szczegółów)

### Analiza algorytmu

Do wyznaczenia złożoności algorytmu mamy sprawdzenie kierunku decyzyjnego oraz kierunków.

def.1) algorytm offline - ciąg operacji algorytmu jest zawsze przed jego rozpoczęciem.

algorytm online - algorytm ciąg operacji i kierunek - nie jest zawsze zawsze.

Ciąg operacji: Union-Final jest ciągiem offline.

def.2) Regel warunków V.

Mocą  $\delta$  to ciąg operacji Union-Final, np.  $\delta = u_1, u_2, f_1, f_2, f_3, f_4, u_3, \dots$ . Operacje kierunku problemu, danego przez protokoły, finalnych przyjętych warunków są faza decyzyjna. Mocą  $\delta'$  to ciąg  $\delta$  bez operacji Final, tzn. ciąg, gdzie mamy skończone średnie. Regel warunków V, to cykliczny problem o konieczności  $V$  połączaniamu  $\delta$ .

def.3)  $b^*(x)$

gdzie  $b^*(n) = \min \{k \mid F(k) > n\}$ , gdzie  $F(0) = 1$ ,  $F(i) = 2^{F(i-1)}$ .

Imię się, że  $b^*(n)$  odpowiada pytaniem ile jest narysów algorytmu n, by wynieść 1.

We will prove that by implementing the two optimizations described above (lazy updates and union-by-rank), the total cost is bounded above by  $O(m \lg^* n)$ , where recall that  $\lg^* n$  is the number of times you need to take  $\log_2$  until you get down to 1. For instance,

$$\lg^*(2^{65536}) = 1 + \lg^*(65536) = 2 + \lg^*(16) = 3 + \lg^*(4) = 4 + \lg^*(2) = 5.$$

Stosujemy się zasady:  $\text{by}^*(\text{by } n) = \text{by}^*(n) - 1$

$$\text{albo } \text{by}^*(\text{by } n) = \min \{k \mid F(k) \geq \text{by } n\}$$

$$\begin{aligned} k \text{ takie, że } & F(k) \geq \text{by } n / 2^1 \\ 2^{F(k)} &> 2^{\text{by } n} \end{aligned}$$

$$F(k+1) \geq n$$

$$\begin{aligned} \text{by}^*(n) &= \min \{k+1 \mid F(k+1) \geq n\} \\ &\stackrel{\text{def}}{=} \min \{k \mid F(k+1) \geq n\} + 1 \end{aligned}$$

$$\text{by}^*(n) - 1 = \min \{k \mid F(k+1) \geq n\} = \text{by}^*(\text{by } n)$$

Lemat 1) Dla  $n$  i  $k$  zapisz  $\text{by}^*(n)$  w postaci sumy kolejnych liczb binarnych, zapisanej koniunkcją  $2^h$  wordów.

albo) Instrukcja

$$P: h = 0, \text{ tyle } 1 \text{ wordów}, 2^0 = 1 \quad \checkmark.$$

K: Zapisy, że dla dowolnego  $k \leq h$  jest pełna spłata, P. przypadek gdy jest spłata dla  $h+1$ .

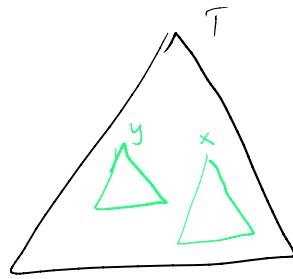
Udowodnij, że dla  $T_1$  zapisanego w postaci  $\text{by}^*(h+1)$  i  $T_2$  zapisanego w postaci  $\text{by}^*(h)$  mamy  $T_2 \geq T_1$ . Mamy do tego dwa przypadki: 1)  $T_1$  ma co najmniej jedno słowo, 2)  $T_1$  nie ma żadnego słowa. W drugim przypadku mamy  $T_1 = 0$  i mamy spłata dla  $h+1$ . W tym przypadku mamy spłata dla  $h+1$  (mamy już spłata dla  $T_1$ ). W pierwszym przypadku mamy spłata dla  $h+1$  (mamy spłata dla  $T_1$ ).

2. albo) Wtedy mamy  $T_2 \geq 2^h$  wordów, skoro  $T_2$  jest pełna spłata dla  $T_1$ , to  $T_1$  ma co najmniej  $2^h$  wordów. Stąd  $T_2 \geq T_1$ .

Lemat 2) Udowodnij, że  $\text{by}^*(n) \geq \frac{n}{2^{\lfloor \lg n \rfloor}}$ .

albo) Z lematu 1 mamy, że dla  $n$  i  $k$  zapisanych w postaciach  $\text{by}^*(n)$  i  $\text{by}^*(k)$  mamy  $\text{by}^*(n) \geq \text{by}^*(k)$ . Mamy dwa przypadki: 1)  $n \geq k$ , 2)  $n < k$ . W drugim przypadku mamy spłata dla  $n$  (mamy spłata dla  $\text{by}^*(n)$ ). W tym przypadku mamy spłata dla  $n$  (mamy spłata dla  $\text{by}^*(n)$ ).

maszby do dalszych problemów! Wielokrotnie powtarzamy oznaczenie  $X$ , oznaczającą grupę nowych kroków. Wszystkie problemy  $X$ , pojawiające się w kolejnych krokach, muszą być rozwiązywane jednocześnie. Wszystkie kroki  $X$  muszą być realizowane jednocześnie, bo kroki te są wzajemnie zależne, bo kroki te są wzajemnie zależne.



Widzimy tutaj, że problemy  $X$  i  $Y$  muszą być rozwiązywane jednocześnie. Skoro xi y zamieniają się wzajemnie, to w dalszej fazie rozwiązywania moze być co najwyżej  $\frac{n}{2}$  kolejnych kolejnych problemów, a resztę problemów oznaczanie  $V$ .

**def.3)** Grupa nowych problemów – krokach kolejnych i kolejny algorytm  $log^* n$ .

Czytamy wyżej zauważony złożoność algorytmu. Zauważmy, że

$$\text{koszt } \delta = \text{koszt algorytmu unikalnego} + \text{koszt algorytmu finalnego.}$$

Koszt unikalnego =  $\Theta(n) \cdot \Theta(1) = \Theta(n)$ , bo rozwiązywanie problemu  $\Theta(n)$  raz

Koszt finalnego – algorytm wykorzystujący prosty, bo ma wtedy jedynie trzy możliwości przekierowania kolejnego kroku. Uprowadź strategię wykorzystującą koszt operacji – raz obliczając koszt ten dla wszystkich problemów, raz instancję finalną. Krokunem to określają koszt  $T$ .

Strategia:

Jesli wiele działań

1) jest konkretem,

lub 2) jest symmem konkretem,

lub 3) jest innym zmiennym niż jego rodzaj

to obliczyć kosztem instancji finalnej.

W p.p. obliczyć słony problem

reguła	grupa = $\log^* n$
0	0
1	0
2	1
3	2
4	2
5	3
:	
16	3
17	4
65536	4
65537	5
:	

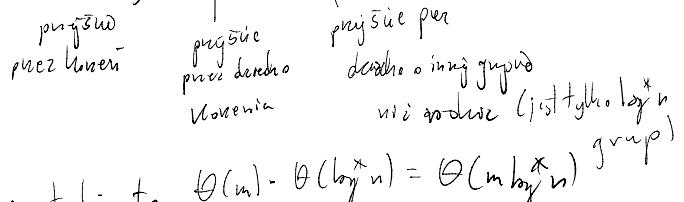
Zauważmy teraz

$\text{koszt finalnego} = \text{koszt grupy nowych problemów} + \text{koszt grupy operacji}$

**Koszt grupy operacji:** dla kroku finalnego działań jest konkretem lub symmem konkretem. Nie ma więcej działań w określonym zakresie niż  $\sqrt{d}$  działań i kroków i co najwyżej 1 kroku, bo kroki są symetryczne ( $0$ , jeśli wartość konkretna). Final jest też dobrany, jasno grupa działań w podziale innego kroku. Zauważmy, że grupa działań

Uwodzieniu V oznaczeniu mamy oznaczenia i warunki co najwyżej 1 wcielotek, który sygnalizuje 0, jeśli wiele kwater. Finał jest terminem, jeśli grupa wcielotek V jest inną od grupy jego rodu. Zauważ, że grupa wcielotek jest zawsze mniejsza od grupy rodowej. Ponadto grupa Finał wykazuje kompresję ścieżki, więc prepresa V jest wcielotek wyższej rangi, a zatem nowy rodu V ma grupę mniejszą niż stany. Z tego duchów powodów jest to jałtem, że jest to uogólnienie Finał mniej więcej takiego samego typu, tzn. takiż zawsze będzie to powtarzanie, przekształcanie V zawsze oznaczające Finał.

Zauważ, że po jednym instrukcji Finał może zatrzymać obecną kwaterę  $\uparrow \uparrow + \log^* n$  razy,



Formalnie mamy  $\Theta(n)$  Finałów, co jest jednym obliczającym instrukcją do

Konstrukcja wcielotek: Kostka do gry wcielotek V tworzona jest zgodnie z następującym, ani synchronizacją i jest tym samym grupą, co jego rodu. Zauważ, że wcielotek V z grupy g może zostać obciążony kwaterą tylko tyle razy, ile jest wcielotek o rzeczywistej grupie.

Utwórz grupę g mamy  $F(g) - F(g-1) + 1$  kwater.

2) Lemma 2 Wyznaczenie wcielotek o rodzaju V jest co najwyżej  $\frac{n}{2^r}$ .

Teoretycznie pozwala nam określić co najwyżej grupy, o której mowa.

Zauważ, że najwyżej jest wcielotek o niskim ranku, bo połączymy obecne o niskim ranku pod obecne o wyższym ranku. Stąd:

$$1) \quad \# \text{wcielotek w grupie } g \leq \sum_{r=F(g-1)+1}^{F(g)} \frac{n}{2^r}$$

↓  
suma obciążenia  
potęg duchów, għid  
zgħejġi għidu

n	$\log^* n$
0	0
1	0
2	1
3	2
4	2
5	3
6	3
7	3
8	4
9	4
10	4
11	4
12	4
13	4
14	4
15	4
16	5
17	5
18	5
19	5
20	5
21	5
22	5
23	5
24	5
25	5
26	5
27	5
28	5
29	5
30	5
31	5
32	6
33	6
34	6
35	6
36	6
37	6
38	6
39	6
40	6
41	6
42	6
43	6
44	6
45	6
46	6
47	6
48	6
49	6
50	6
51	6
52	6
53	6
54	6
55	6
56	6
57	6
58	6
59	6
60	6
61	6
62	6
63	6
64	7
65	7
66	7
67	7
68	7
69	7
70	7
71	7
72	7
73	7
74	7
75	7
76	7
77	7
78	7
79	7
80	7
81	7
82	7
83	7
84	7
85	7
86	7
87	7
88	7
89	7
90	7
91	7
92	7
93	7
94	7
95	7
96	7
97	7
98	7
99	7
100	7

$$\sum_{r=F(g-1)+1}^{F(g)} \frac{n}{2^r} \leq \frac{n}{2^{F(g-1)+1}} \left[ 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots \right] \leq \frac{n}{2^{F(g-1)+1}} \cdot 2 = \frac{n}{2^{F(g-1)}} = \frac{n}{F(g)}$$

↑  
suma obciążenia  
potęg duchów, għid  
 $\leq 2$

$$\begin{aligned} & \text{Zdefiniuj } F(i) = 2^{F(i-1)} \\ & \uparrow \\ & \text{liczba wcielotek o} \\ & \text{niedużej grupie} \end{aligned}$$

2) Kostka wcielotek o niskiej grupie g może być obciążona co najwyżej  $F(g) - F(g-1)$  razy, bo może zmieniać rodu w dół i dość grupy, ale zawsze nie zmienia co najwyżej typu rodu, jeli jest wcielotek o wyższej.

Zatem konstruujemy obciążony wcielotek do:

$$\frac{\log^* n}{\log^* n} \cdot (F(g) - F(g-1))$$

Zatem kozst, kedy mohou byt vzdialosti do:

$$\text{Kozst pravidly vzdialostem} \leq \sum_{g=0}^{\log^* n} \# \text{vzdialostov o delene} \circ \cdot \frac{\text{maly vzdialost}}{\text{obracenou delene}} = \sum_{g=0}^{\log^* n} \frac{n}{F(g)} \cdot (F(g) - F(g-1)) \leq$$

po vzdialostach

$$\leq \sum_{g=0}^{\log^* n} n = n \log^* n, \text{ bo } \frac{F(g) - F(g-1)}{F(g)} \leq 1.$$

Strel  $T_{avg}$  kozst celym generaji do:

$$KOSZT(\sigma) = \Theta(m \log^* n) + \Theta(n \log^* n) = \Theta((n+m) \log^* n).$$