პრაქტიკული 12: ღია მისამართებით ჰეშირება: წრფივი გადასინჯვის მეთოდი

სააუდიტორიო სამუშაო:

- ზოგადი ცნობები და ელემენტის ჩამატების ფსევდოკოდი
- ამოცანა 1 /ჰეშ-ცხრილში ელემენტების ჩასმა/ >>>
- ამოცანა 2 /ელემენტის წაშლა/ >>>
- სავარჯიშოები >>>

ზოგადი ცნობები და ელემენტის ჩამატების ფსევდოკოდი. ჰეშირება არის ასოცირებული კონტეინერების შექმნის ალტერნატიული გზა. ამ შემთხვევაში, ობიექტის საშუალებით გამოითვლება კონტეინერში მისი მისამართი. თუ ჰეშირების ალგორითმი და ჰეშ-ცხრილის ზომები სწორედაა შერჩეული, ელემენტის ჩასმის, წაშლის და მოძებნის დრო არის **მუდმივი.** მეტი ოპერაციები ჰეშირების საშუალებით შექმნილ კონტეინერებს არ მოეთხოვებათ.

ჩვენი ამოცანაა გასაღებები (უნიკალური ნატურალური რიცხვები) შევინახოთ ერთგანზომილებიან ცხრილში გასაღებების სიდიდის გათვალისწინებით (ცხრილის დაპროგრამებისთვის გვჭირდება სწრაფი წვდომის კონტეინერი). ცხრილში უჯრების ნომრებს ვუწოდოთ მისამართები. ცხრილის გადანომრვა იწყება ნულიდან. ცხრილი აღვნიშნოთ T-თი, მისი ზომა აღვნიშნოთ m-ით.

იმისათვის, რომ ელემენტის ჩამატება, ძებნა და ელემენტის წაშლა იყოს შეძლებისდაგვარად სწრაფი, ყოველი გასაღების შესაბამისი მისამართი უნდა გამოვთვალოთ რაიმე ფორმულით, და შემდეგ ამ მისამართზე ჩავწეროთ გასაღები.

ვიგულისხმოთ, რომ აქტიური გასაღებების რაოდენობა ყოველ მომენტში უფრო მცირეა, ვიდრე ცხრილის ზომა, ამუ ყოველთვის არის ღია მისამართები (აქედან მოდის სახელი - ღია მისამართებით ჰეშირება). მოვიყვანოთ ღია მისამართის განსაზღვრის რამდენიმე მეთოდი.

პირველ რიგში, გასაღებების რაოდენობის სიმცირე არ ნიშნავს თვითონ გასაღების სიმცირეს. პირიქით, რადგან გასაღები უნიკალური უნდა იყოს, გამოყენებებში მისი მნიშვნელობა ბევრად დიდია ვიდრე ცხრილის ზომა. ამიტომ გვჭირდება ფუნქციები, რომლებიც გასაღებების მნიშვნელობებს გარდაქმნის {0, 1, ..., m-1} სიმრავლის ელემენტებად, ანუ ამ ცხრილის მისამართებად. ტრადიციულად, ჰეშ ფუნქციის არგუმენტებს **გასაღებებს,** ხოლო კონკრეტულ გასაღებებზე ჰეშ ფუნქციის მნიშვნელობებს **ჰეშ-მნიშვნელობებს** ვუწოდებთ. გასაღებები უნიკალურია, ანუ განსხავებულია ერთმანეთისაგან, მაგრამ ზოგიერთი ჰეშ-მნიშვნელობა აუცილებლად აღმოჩნდება ერთმანეთის ტოლი (ძალიან ბევრი გასაღებია და მხოლოდ m ცალი ჰეშ-მნიშვნელობა). ჰეშ მნიშვნელობების დამთხვევას ეწოდება **კოლიზია.** ჰეშირების პროცესში, ღია მისამართის მოძებნის დროს კოლიზები ქმნიან ძირითად სითულეს.

განვიხილოთ ორი სახის ჰეშ-ფუნქცია: ჰეშირება გაყოფით და გამრავლებით.

გაყოფის მეთოდი ერთ ჰეშ-ფუნქციას გვაძლევს: $h(k) = k \% m$;

გამრავლების მეთოდში მონაწილეობს $0 < A < 1$ პარამეტრი. პარამეტრის ყოველ მნიშნელობას შეესაბამება ერთი ჰეშ-ფუნქცია: $h(k) = \lfloor m \times ((k \cdot A) \bmod 1) \rfloor$. ხშირად იღებენ:

$$A \approx (\sqrt{5} - 1)/2 = 0.6180339887.$$

ცხადია, ჰეშ-ფუნქციის განსაზღვრაში მონაწილეობს m სიდიდე, მაგრამ რადგან ყოველი ცხრილისთვის ეს სიდიდე მუდმივია, ამიტომ სიმარტივისთვის მას აღარ მიუთითებენ $h(k)$ აღნიშვნაში.

რადგან რამდენიმე გასაღებს შეიძლება შეესაბამებოდეს ჰეშ-ფუნქციით გამოთვლილი ერთი მისამართი (ე.ი. დგილი აქვს კოლიზიას), მაგრამ ცხრილში თვისუფალი ანუ ღია მისამართი ყოველთვის არსებობს, ამიტომ ვიქცევით შემდეგნაირად: თუ $h(k)$ დაკავებულია, ვცდილობთ k გასაღები ჩავწეროთ მის მომდევნო მისამართზე, რომელიც გამოითვლება $(h(k)+1)\%m$ წესით. თუ ესეც დაკავებულია, მაშინ ამის მომდევნოზე $(h(k)+2)\%m$, და ა.შ., ვიდრე არ მივაგნებთ თავისუფალ მისამართს. თუ m მისამართის გასინჯვის შემდეგ ღია მისამართი ვერ ვნახეთ, ეს ნიშნავს რომ ცხრილი მთლიანად შევსებულია, რაც დაშვების თანახმად არ უნდა მომხდარიყო და ამიტომ გვაქვს გადავსების შეცდომა. რადგან ღია მისამართის ძებნაში მისამართებს ვსინჯავთ მიმდევრობით. მეთოდს ჰქვია **ღია მისამართების განსაზღვრა წრფივი გადასინჯვის მეთოდით**. თავისი სიმარტივის გამო, ზოგჯერ იქმნება დაკავებლი უჯრების გრძელი სერიები, რასაც კლასტერები ეწოდება და რაც ანელებს ძებნას.

არსებობს ღია მისამართის განსაზღვრის სხვა მეთოდებიც. მათგან აღსანიშნავია ე.წ. ორმაგი ჰეშირება. როდესაც i-ურ ცდაზე ვცდილობთ k გასაღების ჩამატებას $\left(h_1(k)+ih_2(k)\right)\%m$ მისამართზე, სადაც $h_1(k), h_2(k)$ ორი ჰეშ-ფუნქციაა (რომელთაგან მეორე აუცილებლად ნულისგან განსხვავებულ მნიშვნელობებს უნდა იძლეოდეს).

სხვადასხვა მეთოდის განზოგადებისთვის, მოსახერხებელია ჰეშ-ფუნქციისთვის ისეთი აღნიშვნის შემოღება, რომელიც მოიცავს გასაღებსაც და ცდის ნომერსაც. ასეთი აღნიშვნაა $H(k,i)$. მაგალითად, წრფივი გასინჯვების მეთოდში

$$H(k,i) = (h(k)+i)\%m,$$

ხოლო ორმაგი ჰეშირების შემთხვევაში

$$H(k,i) = \left(h_1(k)+ih_2(k)\right)\%m.$$

ცხრილში ელემენტის ჩამატების ალგორითმის ფსევდოკოდი ასეთია ცარიელი ცხრილი შევსებულია ნულებით, ხოლო რომელიმე ელემენტის წაშლის დროს ვწერთ -1-ს):

- მოცემული k გასაღებისთვის ვიპოვოთ ჰეშ-მნიშვნელობა h(k);
- დაწყებული i=0 -იდან, დავიწყოთ ცხრილის უჯრების გადასინჯვა მანამდე, ვიდრე არ შეგვხვდება თავისუფალი უჯრა, ანუ უჯრა რომელშიც წერია 0 ან -1. ყოველი i -ური ცდისთვის მისამართს გამოვთვლით ფორმულით: *(h(k)+i)mod(m)*, ანუ ჯერ გამოვთვლით h(k) მისამართს, მერე, თუ იგი დაკავებულია, მის მარჯნივ პირველივე თვისუფალს ვეძებთ.
- თუ რომელიმე i -ური ცდისთვის (h(k)+i)mod m მისამართზე წერია ნული ან -1, მაშინ ამ უჯრაში ჩავწერთ k-ს.
- თუ i გახდა m-ის ტოლი მაგრამ თავისუფალი უჯრა ვერ ვნახეთ, მაშინ გვაქვს ცხრილის გადავსების შეცდომა.

უფრო დეტალურად:

```
int insert(int k)
{
   for(int i=0; i<m; i++)
   {
        int j = H(k,i);
        if( T[j]<=0)
        {
            T[j] = k;
            size++;
            return j;
        }
   }
   cout << "Hash Table Overflow!" << endl;
   return m;
}
```

<<< **ამოცანა 1** /ჰეშ-ცხრილში ელემენტების ჩასმა/. თავდაპირველად ცარიელ 11-ელემენტიან ჰეშ ცხრილში ჩავამატოთ გასაღებები 10, 22, 31, 4, 15, 28, 17, 88, 59 წრფივი გადასინჯვით ღია მისამართების განსაზღვრის მეთოდით. ჰეშ ფუნქცია ავიღოთ გაყოფის მეთოდით.

ამოხსნა: პირველ რიგში ვნახოთ თუ რა მნიშვნელობებს იღებენ მარტივი ჰეშ-ფუნქციები ამ გასაღებებზე m =11 მნიშვნელობისთვის.

k	k%11	$\left\lfloor 11\cdot\left(\left(\frac{\sqrt{5}-1}{2}k\right)\bmod 1\right)\right\rfloor$
10	10	1
22	0	6
31	9	1
4	4	5
15	4	2
28	6	3
17	6	5
88	0	4
59	4	5

ჰეშ ფუნქცია ავიღოთ გაყოფის მეთოდით, ღია მისამართები განვსაზღვროთ წრფივი გადასინჯვით. ე.ი.

$$H(k,i) = (k\%11+i)\%m,$$

ჩავამატოთ k =10. $H(10,0) = (10\%11+0)\%10 = 10$, ეს მისამართი ღიაა და გასაღები ჩაჯდება:

0	1	2	3	4	5	6	7	8	9	10

0	0	0	0	0	0	0	0	0	0	10

ჩავამატოთ k=22. $H(22,0)=0$, ეს მისამართი ღიაა და გასაღები ჩაჯდება:

0	1	2	3	4	5	6	7	8	9	10
22	0	0	0	0	0	0	0	0	0	10

ჩავამატოთ k=31. $H(31,0)=9$, ეს მისამართი ღიაა და გასაღები ჩაჯდება:

0	1	2	3	4	5	6	7	8	9	10
22	0	0	0	0	0	0	0	0	31	10

ჩავამატოთ k=4. $H(4,0)=4$, ეს მისამართი ღიაა და გასაღები ჩაჯდება:

0	1	2	3	4	5	6	7	8	9	10
22	0	0	0	4	0	0	0	0	31	10

ჩავამატოთ k=15. $H(15,0)=4$, ეს მისამართი დაკავებულია; $H(15,1)=5$, ეს მისამართი ღიაა და გასაღები ჩაჯდება:

0	1	2	3	4	5	6	7	8	9	10
22	0	0	0	4	15	0	0	0	31	10

ჩავამატოთ k=28. $H(28,0)=6$, ეს მისამართი ღიაა და გასაღები ჩაჯდება:

0	1	2	3	4	5	6	7	8	9	10
22	0	0	0	4	15	28	0	0	31	10

ჩავამატოთ k=17. $H(17,0)=6$, ეს მისამართი დაკავებულია; $H(17,1)=7$, ეს მისამართი ღიაა და გასაღები ჩაჯდება:

0	1	2	3	4	5	6	7	8	9	10
22	0	0	0	4	15	28	17	0	31	10

ჩავამატოთ k=88. $H(88,0)=0$, ეს მისამართი დაკავებულია; $H(88,1)=1$, ეს მისამართი ღიაა და გასაღები ჩაჯდება:

0	1	2	3	4	5	6	7	8	9	10
22	88	0	0	4	15	28	17	0	31	10

ჩავამატოთ k=59. $H(59,0)=4$, ეს მისამართი დაკავებულია; $H(59,1)=5$ დაკავებულია და ა.შ. მხოლოდ $H(59,4)=8$ არის ღიაა:

0	1	2	3	4	5	6	7	8	9	10
22	88	0	0	4	15	28	17	59	31	10

<<< ამოცანა 2 /ელემენტის წაშლა/. ამოცანა 1-ის ჰეშ-ცხრილში წავშალოთ ელემენტი 15, მერე წავშალოთ 59.

ამოხსნა: პირველ რიგში ვნახოთ ძებნის და წაშლის ალგორითმის ფსევდოკოდები:

```
int search(int k)
{
    int i = 0;
    while( true)
    {
          int j = H(k,i);
          if( T[j] == k)
               return j;
          ifT[j] == 0 || i == m)
               return m;
          i++;
    }
}
void erase(int k)
{
     int j =  search(k);
     if( j != m )
     {
           T[j] = -1;
           size--;
     }
}
```

წაშლის ფუნქცია თავიდან იძახებს ძებნას.

როცა `i = 0`, მაშინ `j = H(15,0)` = 4. `T[4] = 4`, რაც განსხვავდება საძებნი ელემენტისგან, ნულისგან და ცხრილის ზომისგან, ამიტომ ვიღებთ `i = 1`. ამ შემთხვევაში `j = H(15,1)` = 5. `T[5]` = 15 და

```
int j =  search(15);
```

შეტყობინების შედეგად `j` = 5. ახლა, რადგან `if( j != m )` ჭეშმარიტია, ამიტომ `T[5]= -1;` და ცხრილის ზომაც (ელემენტების რაოდენობა) მცირდება ერთით.

ანალოგიურად მივიღებთ, რომ `T[8] = -1`.

წაშლის ალგორითმი წაშლილ მისამართზე 0-ს რომ წერდეს, მაშინ პრობლემები შეიქმნებოდა ძებნასთან დაკავშირებით, 59-ის ძებნის ფუქნცია დააბრუნებდა ცხრილის ზომას (იტერატორების ენაზე ეს არის end()), და 59 დარჩებოდა ცხრილში.

<<< სავარჯიშოები

1. ამოხსენით იგივე ამოცანა, ოღონდ ჰეშ-ფუნქცია აიღეთ გამრავლების მეთოდით (ჰეშ-მნიშვნელობები მოცემულია ცხრილში).
2. თავდაპირველად ცარიელ 13-ელემენტიან ჰეშ ცხრილში ჩაამატეთ გასაღებები 10, 22, 31, 4, 15, 28, 17, 88, 59 77 და 40, ღია მისამართები განსაზღვრეთ წრფივი გადასინჯვით. ჰეშ ფუნქცია აიღეთ გაყოფის მეთოდით.
3. წინა სავარჯიშოში აგებულ ცხრილში, წაშალეთ ელემენტი 15, შემდეგ 59.
4. წინა ორ სავარჯიშოში ჰეშ-ფუნქცია აიღეთ გამრავლების მეთოდით.