12-§. Hisoblash geometriyasi algoritmlari

Hisoblash geometriyasi - geometrik masalalarni yechish algoritmlari bilan shug‘ullanadigan informatika bo'limi.

Bu uchburchak, qavariq sirtlarni qurish, bitta obyektning boshqasiga tegishliligini aniqlash, ularning kesishishini topish va boshqalar kabi vazifalar bilan shug‘ullanadi, ular geometrik obyektlar bilan ishlaydi: nuqta, segment, ko'pburchak, aylana va hokazolar.

Hisoblash geometriyasi kompyuter grafikalarida, muhandislik dizaynida va boshqa ko‘plab geometriya sohalarida qo'llaniladi.

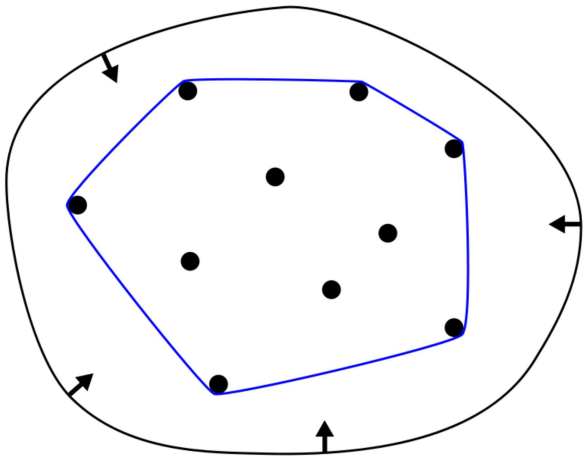
12.1. Qavariq qobiq muammolari

X to'plamining qavariq qobig‘i - X ni o'z ichiga olgan eng kichik qavariq to'plami. "Eng kichik to'plam" bu yerda to'plamlarni joylashtirishga nisbatan eng kichik elementni, ya'ni berilgan raqamni o'z ichiga olgan shunday qavariq to'plamni anglatadiki, u berilgan figurani o'z ichiga olgan boshqa har qanday qavariq to'plamda mavjud.

X to'plamining qavariq tanasi odatda ConvX bilan belgilanadi.

Misol. Ko'plab mixlar mixlangan taxtani tasavvur qiling. Arqonni oling, ustiga sirpanchiq ilmoq (lasso) bog‘lab, taxtaga tashlang va keyin mahkamlang. (54-rasm)

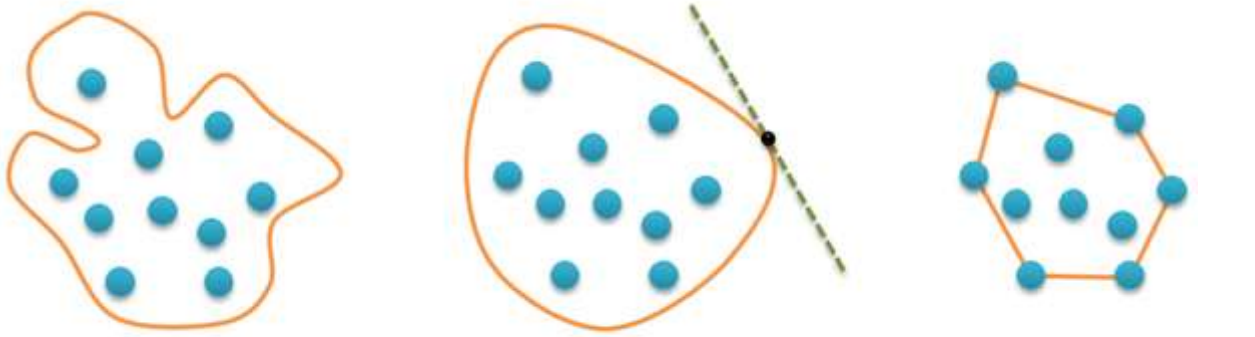
Arqon barcha mixlarni o'rab oladi, lekin u faqat eng tashqi qismlariga tegadi. U tegib turgan mixlar butun mixlar guruhi uchun qavariq qobiqni hosil qiladi.



54-rasm. Ko‘plab mixlar mixlangan taxta tasviri

Minimal qavariq qobiq tushunchasi. Tekislikda cheklangan A nuqtalar to'plami berilgan bo'lsin. Bu to'plamning konvertlari o'zaro kesishmalarsiz har qanday yopiq H chiziq bo'lib, A ning barcha nuqtalari shu egri chiziq ichida yotadi. Agar H egri chiziq qavariq bo'lsa (masalan, bu egri chiziqning har qanday urinish nuqtasi uni boshqa biron bir nuqtada kesib o'tmasa), u holda tegishli qobiq ham qavariq deb ataladi. Va nihoyat, minimal qavariq qobiq minimal uzunlikdagi (minimal perimetr) qavariq qobiq deb ataladi. Barcha kiritilgan tushunchalar quyidagi 55-rasmda keltirilgan.

A nuqtalar to'plamli minimal qavariq qobiqning asosiy xususiyati shundaki, bu tanasi qavariq ko'pburchak bo'lib, uning uchlari A dagi bir nechta nuqtadir, shuning uchun minimal qavariq qobiqni topish muammosi oxir-oqibat A dan kerakli nuqtalarni tanlash va tartiblashgacha kamayadi. Algoritm chiqishi ko'pburchak bo'lishi kerakligi sababli tartiblash ya’ni saralash zarur, ya'ni uchlar ketma-ketligi bo‘yicha. Uchlar tartibiga qo'shimcha ravishda shart qo'yamiz - ko'pburchakning o'tish yo'nalishi musbat bo'lishi kerak (soat strelkasi bo‘yicha).



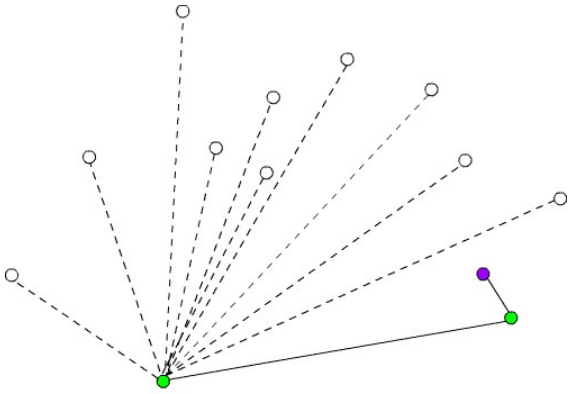
a) Qobiq **b) Qavariq qobiq** **c) Minimal qavariq qobiq**

55-rasm. Qobiq, qavariq qobiq va minimal qavariq qobiq tushunchalari

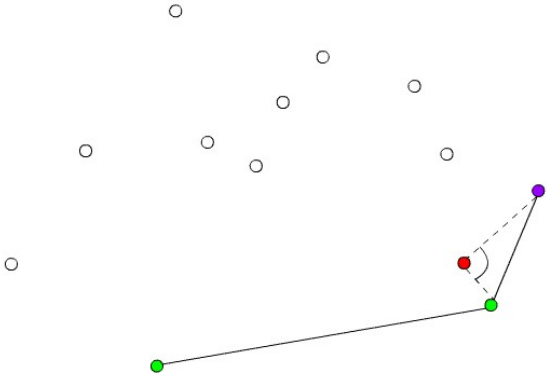
Minimal qavariq qobiqni qurish masalasi hisoblash geometriyasidagi eng oddiy muammolardan biri hisoblanadi; buning

uchun juda ko'p turli algoritmlar mavjud. Quyida biz ikkita algoritmni ko'rib chiqamiz – Grexem (Graham scan) va Jarvis (Jarvis march).

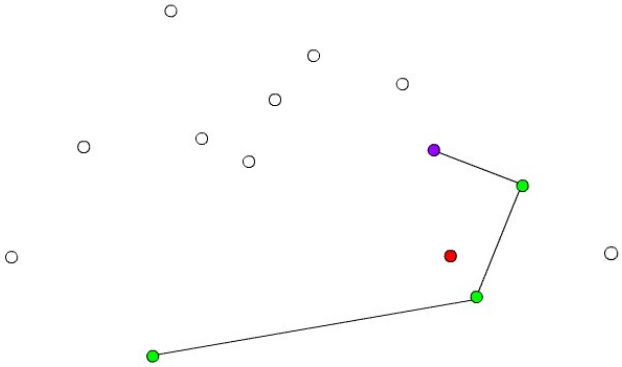
Grexem algoritmi quyidagi 56-rasmda ketma-ket keltirilgan.



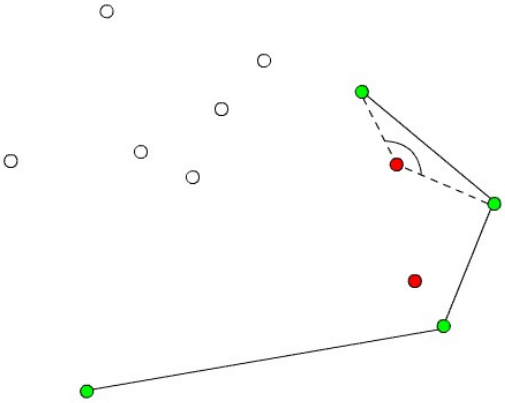
1-qadam



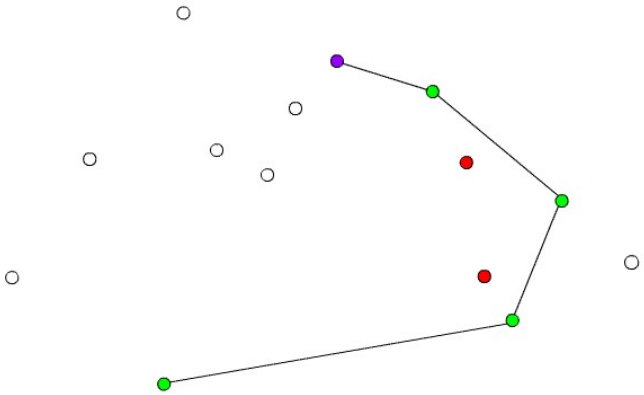
2-qadam



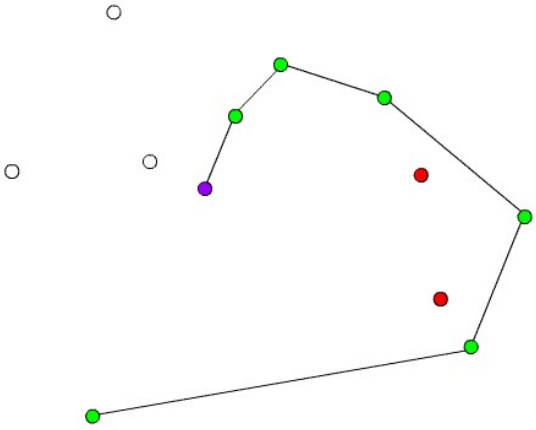
3-qadam



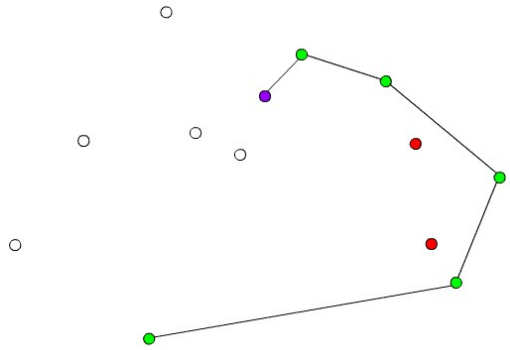
4-qadam



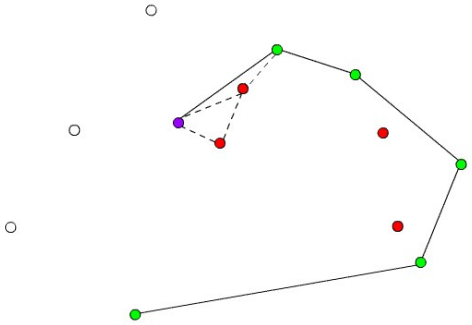
5-qadam



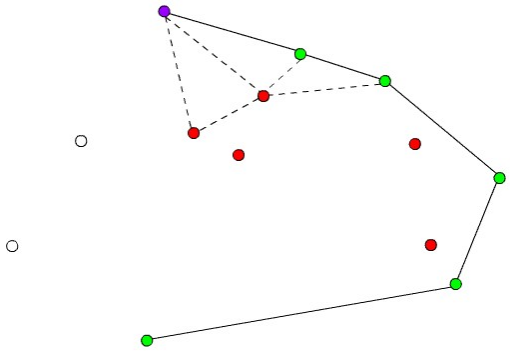
6-qadam



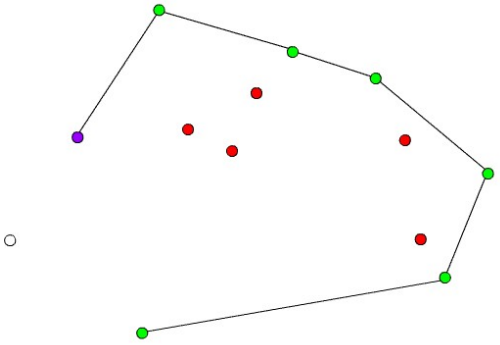
7-qadam



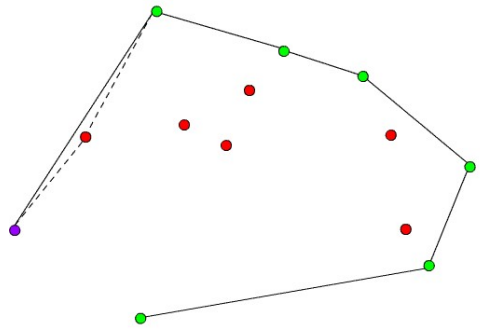
8-qadam



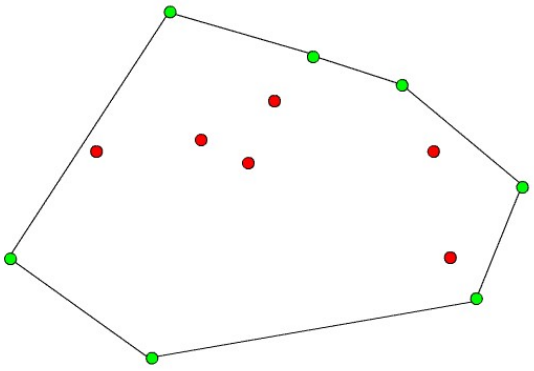
9-qadam



10-qadam



11-qadam



12-qadam

56-rasm. Grexem algoritmining bajarilish ketma-ketligi

Grexem algoritmi realizatsiyasi (C++ tilida)

```
#include <iostream>
#include <stack>
#include <stdlib.h>
using namespace std;

struct Point
{
	int x, y;
};

Point p0;
Point nextToTop(stack<Point> &S)
{
	Point p = S.top();
	S.pop();
	Point res = S.top();
	S.push(p);
	return res;
}
void swap(Point &p1, Point &p2)
{
	Point temp = p1;
	p1 = p2;
	p2 = temp;
}
int distSq(Point p1, Point p2)
{
	return (p1.x - p2.x)*(p1.x - p2.x) +
		(p1.y - p2.y)*(p1.y - p2.y);
}
int orientation(Point p, Point q, Point r)
{
	int val = (q.y - p.y) * (r.x - q.x) -
			(q.x - p.x) * (r.y - q.y);

	if (val == 0) return 0;
```

```
        return (val > 0)? 1: 2;
}
int compare(const void *vp1, const void *vp2)
{
Point *p1 = (Point *)vp1;
Point *p2 = (Point *)vp2;


int o = orientation(p0, *p1, *p2);
if (o == 0)
        return (distSq(p0, *p2) >= distSq(p0, *p1))? -1 : 1;

return (o == 2)? -1: 1;
}
void convexHull(Point points[], int n)
{

int ymin = points[0].y, min = 0;
for (int i = 1; i < n; i++)
{
        int y = points[i].y;

        if ((y < ymin) || (ymin == y &&
                points[i].x < points[min].x))
                ymin = points[i].y, min = i;
}
swap(points[0], points[min]);
p0 = points[0];
qsort(&points[1], n-1, sizeof(Point), compare);
int m = 1; // Initialize size of modified array
for (int i=1; i<n; i++)
{

        while (i < n-1 && orientation(p0, points[i],
points[i+1]) == 0)
                i++;
        points[m] = points[i];
        m++;
if (m < 3) return;
```

```
    stack<Point> S;
    S.push(points[0]);
    S.push(points[1]);
    S.push(points[2]);


    for (int i = 3; i < m; i++)
    {

        while (S.size()>1 && orientation(nextToTop(S), S.top(), points[i])
!= 2)
            S.pop();
        S.push(points[i]);
    }
    while (!S.empty())
    {
        Point p = S.top();
        cout << "(" << p.x << ", " << p.y <<")" << endl;
        S.pop();
    }
    }
    }
    int main()
    {
        Point points[] = {{0, 3}, {1, 1}, {2, 2}, {4, 4},
                          {0, 0}, {1, 2}, {3, 1}, {3, 3}};
        int n = sizeof(points)/sizeof(points[0]);
        convexHull(points, n);
        return 0;
    }
```

12.2. Tekislikda chiziqlar kesishgan sohalarni qidirish algoritmi (Sweep Line)

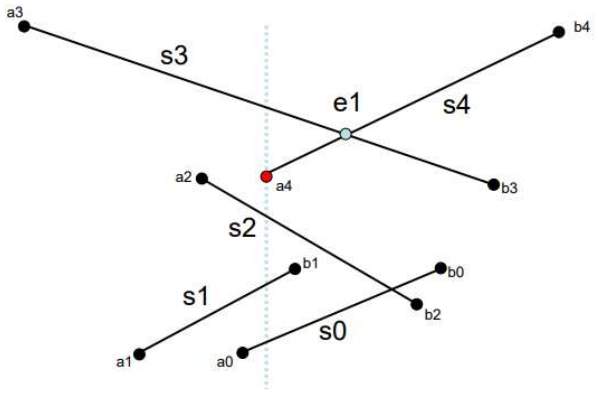
Tekislikda chiziqlar kesishgan sohalarni qidirish algoritmi(Sweep Line) - bu Yevklid fazosidagi turli xil muammolarni hal qilish uchun kesishgan sohalardan foydalanadigan algoritmik paradigma. Bu hisoblash geometriyasidagi asosiy texnikalardan biridir.

Ushbu turdagi algoritmlarning g‘oyasi ba'zi bir nuqtalarda to'xtab, tekislik bo'ylab harakatlanadigan xayoliy to'g‘ri chiziqni (ko'pincha vertikal) tasavvur qilishdir. Geometrik amallar, kesishgan chiziqqa qo'shni bo'lgan geometrik obyektlar bilan cheklanadi va chiziq barcha obyektlardan o'tib ketganda to'liq yechim mavjud bo‘ladi.

Sweep Line – bu tekislik bo'ylab to'g‘ri yo'nalishda siljigan xayoliy vertikal chiziq. Shuning uchun bu konsepsiyaga asoslangan algoritmlarni ba'zan tekisliklarni tozalash algoritmlari deb ham atashadi. Chiziqni diskretlashtirish uchun biz ba'zi hodisalarga asoslanib tozalaymiz.

Yana shuni ta'kidlash kerakki, bu texnikaning samaradorligi biz foydalanadigan ma'lumotlar tuzilmalariga bog‘liq. Umuman olganda, biz C++ da setdan foydalanishimiz mumkin, lekin ba'zida biz qo'shimcha ma'lumotlarni saqlashni talab qilamiz, shuning uchun biz muvozanatlashgan ikkilik daraxtga o'tamiz.

Misol. Sweep Line algoritmining ishlash tartibi



Sweep Line holati (SLH): s0, s1, s2, s3

Navbat (N): a4, b1, b2, b0, b3, b4

Harakatlar:

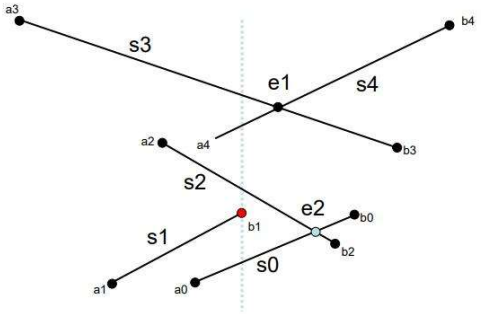
SLH -ga s4 -ni kiriting

Test s4-s3 va s4-s2

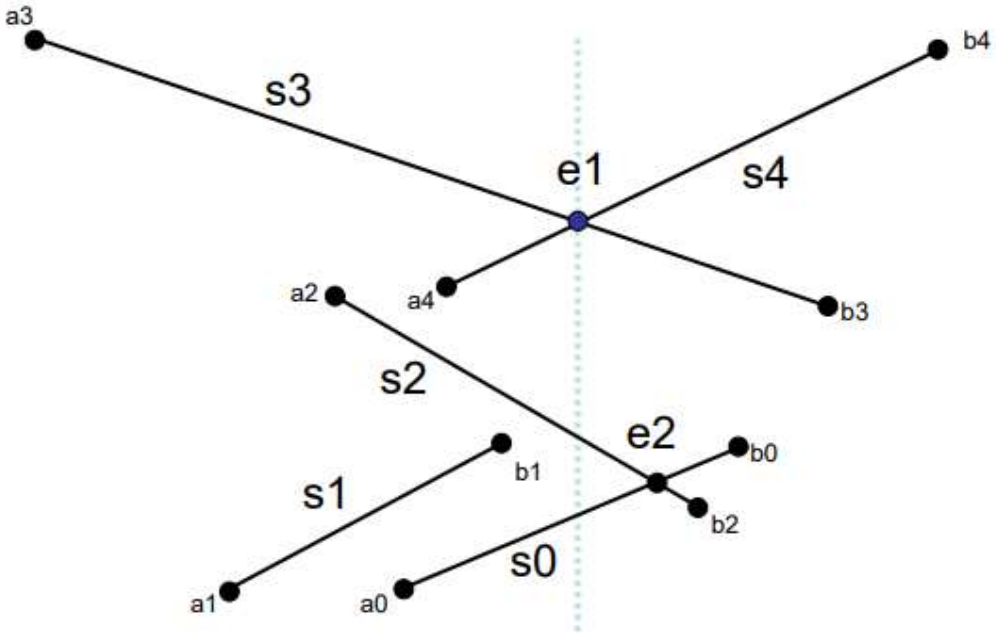
N ga e1 ga qo'shing

Sweep Line holati: s0, s1, s2, s4, s3

Navbat: b1,e1, b2, b0, b3, b4



Harakatlar:
SLH dan s1 ni o‘chiring
Test s0-s2
N ga e2 ga qo'shing
Sweep Line holati: s0, s2, s4, s3
Navbat: e1, e2, b2, b0, b3, b4



Harakatlar:
SLH da s3 va s4 ni almashtiring
Test s3-s2

Sweep Line holati: s0, s2, s3, s4
Navbat: e2, b2, b0, b3, b4

Mavzu yuzasidan savollar:

1. Hisoblash geometriya qanday masalalarni hal qiladi?
2. Yuqoridagi keltirilgan hisoblash geometriya masalalaridan tashqari qanday masalalarni keltira olasiz?
3. Minimal qavariq qobiq tushunchasiga ta’rif bering.

13-§. Xesh jadvallar

13.1. Xesh jadvallar va ularni tashkil etish

Xesh jadvali - bu assotsiativ massiv interfeysini amalga oshiruvchi ma'lumotlar tuzilmasi, ya'ni juftlarni saqlashga (kalit, qiymat) va uchta amalni bajarishga imkon beradi: yangi juftlikni qo'shish, qidirish amali va juftlikni kalit bilan o'chirish.

Xesh jadvallarining ikkita asosiy varianti mavjud: zanjirli va ochiq adreslash. Xesh jadvali ba'zi bir *H* massivini o'z ichiga oladi, ularning elementlari juftliklar (ochiq adreslash bilan xesh jadvali) yoki juftliklar ro'yxati (zanjir bilan xesh jadvali) bo‘ladi.

Xeshlash – bu ixtiyoriy uzunlikdagi kirish ma'lumotlari majmuasini ma'lum bir algoritm tomonidan bajarilgan, belgilangan o'lchamdagi chiqish massiviga aylantirish jarayoni. Bunday algoritmni amalga oshiruvchi funksiya **xesh funksiya**, transformatsiya natijasi xesh yoki xesh yig‘indisi deyiladi. Xesh funksiyasi quyidagi xususiyatlarga ega:

- bir xil ma'lumotlar bir xil xeshni beradi;
- "deyarli har doim" turli xil ma'lumotlar boshqacha xesh beradi.

Ikkinchi xususiyatdagi "deyarli har doim" izohi xeshlarning aniq o'lchamiga ega bo'lishidan kelib chiqadi, shu bilan birga kirish ma'lumotlari bu bilan cheklanmaydi. Natijada, biz xesh funktsiyasi kirish ma'lumotlari to'plamidan xeshlar to'plamiga xaritalashni amalga oshiramiz, bu esa ularning kardinalligi ancha past bo'ladi. Dirixle prinsipiga ko'ra, har bir xesh uchun bir nechta turli xil ma'lumotlar