

# Nálægustu punktar í plani

Bergur Snorrason

30. mars 2022

- ▶ Gefnir eru  $n$  punktar í plani.

- ▶ Gefnir eru  $n$  punktar í plani.
- ▶ Hvaða tveir punktar hafa minnstu fjarlægð sín á milli?

- ▶ Gefnir eru  $n$  punktar í plani.
- ▶ Hvaða tveir punktar hafa minnstu fjarlægð sín á milli?
- ▶ Við getum leyst þetta með tæmandi leit í  $\mathcal{O}(\quad)$ .

- ▶ Gefnir eru  $n$  punktar í plani.
- ▶ Hvaða tveir punktar hafa minnstu fjarlægð sín á milli?
- ▶ Við getum leyst þetta með tæmandi leit í  $\mathcal{O}(n^2)$ .

- ▶ Gefnir eru  $n$  punktar í plani.
- ▶ Hvaða tveir punktar hafa minnstu fjarlægð sín á milli?
- ▶ Við getum leyst þetta með tæmandi leit í  $\mathcal{O}(n^2)$ .
- ▶ Við skoðum einfaldlega öll pör punkta.

- ▶ Gefnir eru  $n$  punktar í plani.
- ▶ Hvaða tveir punktar hafa minnstu fjarlægð sín á milli?
- ▶ Við getum leyst þetta með tæmandi leit í  $\mathcal{O}(n^2)$ .
- ▶ Við skoðum einfaldlega öll pör punkta.
- ▶ Prófum að bæta þetta með því að deila og drottna.

- ▶ Röðum punktunum eftir  $x$ -hniti og skiptum í helminga.



- ▶ Röðum punktunum eftir  $x$ -hniti og skiptum í helminga.
- ▶ Látum  $x_0$  vera þannig að hann liggi á milli  $x$ -hnita helminganna.

- ▶ Röðum punktunum eftir  $x$ -hniti og skiptum í helminga.
- ▶ Látum  $x_0$  vera þannig að hann liggi á milli  $x$ -hnita helminganna.
- ▶ Leysum svo endurkvæmt fyrir hvorn helming fyrir sig.

- ▶ Röðum punktunum eftir  $x$ -hniti og skiptum í helminga.
- ▶ Látum  $x_0$  vera þannig að hann liggi á milli  $x$ -hnita helminganna.
- ▶ Leysum svo endurkvæmt fyrir hvorn helming fyrir sig.
- ▶ Við þurfum nú að athuga hvort eitthvert par á milli helminganna sé betra en bestu pörin í hvorum helmingnum.

- ▶ Röðum punktunum eftir  $x$ -hniti og skiptum í helminga.
- ▶ Látum  $x_0$  vera þannig að hann liggi á milli  $x$ -hnita helminganna.
- ▶ Leysum svo endurkvæmt fyrir hvorn helming fyrir sig.
- ▶ Við þurfum nú að athuga hvort eitthvert par á milli helminganna sé betra en bestu pörin í hvorum helmingnum.
- ▶ Það er of hægt að skoða öll pörin sem liggja á milli, þá verður tímaflækjan  $\mathcal{O}(n^2)$ .

- ▶ Röðum punktunum eftir  $x$ -hniti og skiptum í helminga.
- ▶ Látum  $x_0$  vera þannig að hann liggi á milli  $x$ -hnita helminganna.
- ▶ Leysum svo endurkvæmt fyrir hvorn helming fyrir sig.
- ▶ Við þurfum nú að athuga hvort eitthvert par á milli helminganna sé betra en bestu pörin í hvorum helmingnum.
- ▶ Það er of hægt að skoða öll pörin sem liggja á milli, þá verður tímaflækjan  $\mathcal{O}(n^2)$ .
- ▶ Látum  $d$  vera minnstu fjarlægðina sem fannst í hvorum helmingnum fyrir sig.

- ▶ Röðum punktunum eftir  $x$ -hniti og skiptum í helminga.
- ▶ Látum  $x_0$  vera þannig að hann liggi á milli  $x$ -hnita helminganna.
- ▶ Leysum svo endurkvæmt fyrir hvorn helming fyrir sig.
- ▶ Við þurfum nú að athuga hvort eitthvert par á milli helminganna sé betra en bestu pörin í hvorum helmingnum.
- ▶ Það er of hægt að skoða öll pörin sem liggja á milli, þá verður tímaflækjan  $\mathcal{O}(n^2)$ .
- ▶ Látum  $d$  vera minnstu fjarlægðina sem fannst í hvorum helmingnum fyrir sig.
- ▶ Við getum því hunsað þá punkta sem hafa  $x$ -hnit utan bilsins  $[x_0 - d, x_0 + d]$ .

- ▶ Röðum punktunum eftir  $x$ -hniti og skiptum í helminga.
- ▶ Látum  $x_0$  vera þannig að hann liggi á milli  $x$ -hnita helminganna.
- ▶ Leysum svo endurkvæmt fyrir hvorn helming fyrir sig.
- ▶ Við þurfum nú að athuga hvort eitthvert par á milli helminganna sé betra en bestu pörin í hvorum helmingnum.
- ▶ Það er of hægt að skoða öll pörin sem liggja á milli, þá verður tímaflækjan  $\mathcal{O}(n^2)$ .
- ▶ Látum  $d$  vera minnstu fjarlægðina sem fannst í hvorum helmingnum fyrir sig.
- ▶ Við getum því hunsað þá punkta sem hafa  $x$ -hnit utan bilsins  $[x_0 - d, x_0 + d]$ .
- ▶ Röðum afgangnum eftir  $y$ -hniti.

- ▶ Röðum punktunum eftir  $x$ -hniti og skiptum í helminga.
- ▶ Látum  $x_0$  vera þannig að hann liggi á milli  $x$ -hnita helminganna.
- ▶ Leysum svo endurkvæmt fyrir hvorn helming fyrir sig.
- ▶ Við þurfum nú að athuga hvort eitthvert par á milli helminganna sé betra en bestu pörin í hvorum helmingnum.
- ▶ Það er of hægt að skoða öll pörin sem liggja á milli, þá verður tímaflækjan  $\mathcal{O}(n^2)$ .
- ▶ Látum  $d$  vera minnstu fjarlægðina sem fannst í hvorum helmingnum fyrir sig.
- ▶ Við getum því hunsað þá punkta sem hafa  $x$ -hnit utan bilsins  $[x_0 - d, x_0 + d]$ .
- ▶ Röðum afgangnum eftir  $y$ -hniti.
- ▶ Svo kemur trikkið.



- ▶ Röðum punktunum eftir  $x$ -hniti og skiptum í helminga.
- ▶ Látum  $x_0$  vera þannig að hann liggi á milli  $x$ -hnita helminganna.
- ▶ Leysum svo endurkvæmt fyrir hvorn helming fyrir sig.
- ▶ Við þurfum nú að athuga hvort eitthvert par á milli helminganna sé betra en bestu pörin í hvorum helmingnum.
- ▶ Það er of hægt að skoða öll pörin sem liggja á milli, þá verður tímaflækjan  $\mathcal{O}(n^2)$ .
- ▶ Látum  $d$  vera minnstu fjarlægðina sem fannst í hvorum helmingnum fyrir sig.
- ▶ Við getum því hunsað þá punkta sem hafa  $x$ -hnit utan bilsins  $[x_0 - d, x_0 + d]$ .
- ▶ Röðum afgangnum eftir  $y$ -hniti.
- ▶ Svo kemur trikkið.
- ▶ Okkur nægir, fyrir hvern punkt, að skoða fastann fjölda af næstu punktum.

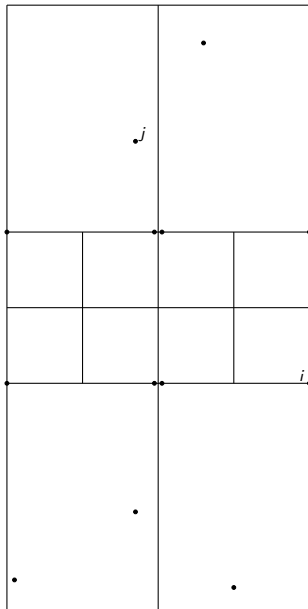
- ▶ Skiptum svæðinu fyrir ofan punktinn  $x_i$  í átta ferninga, hver með hliðarlengdir  $d/2$ .

- ▶ Skiptum svæðinu fyrir ofan punktinn  $x_i$  í átta ferninga, hver með hliðarlengdir  $d/2$ .
- ▶ Ef fjarlægðin milli alla punktana í hvorum helming er ekki minni en  $d$  þá getur mest einn punktur verið í hverjum ferningi (þar með talið er  $x_i$ ).

- ▶ Skiptum svæðinu fyrir ofan punktinn  $x_i$  í átta ferninga, hver með hliðarlengdir  $d/2$ .
- ▶ Ef fjarlægðin milli alla punktana í hvorum helming er ekki minni en  $d$  þá getur mest einn punktur verið í hverjum ferningi (þar með talið er  $x_i$ ).
- ▶ Allir punktar utan þessa svæðis eru meira en  $d$  fjarlægð frá  $i$ -ta punktinum, svo við þurfum ekki að skoða þá.

- ▶ Skiptum svæðinu fyrir ofan punktinn  $x_i$  í átta ferninga, hver með hliðarlengdir  $d/2$ .
- ▶ Ef fjarlægðin milli alla punktana í hvorum helming er ekki minni en  $d$  þá getur mest einn punktur verið í hverjum ferningi (þar með talið er  $x_i$ ).
- ▶ Allir punktar utan þessa svæðis eru meira en  $d$  fjarlægð frá  $i$ -ta punktinum, svo við þurfum ekki að skoða þá.
- ▶ Svo við þurfum bara að skoða fjarlægðina frá  $x_i$  í  $x_j$  þegar  $j - i \leq 7$ .





```

23 double closest_pair_r(pt* a, pt *b, int n, pt* r1, pt* r2)
24 { // Hjálparfall.
25     if (n < 2) return 1e16;
26     if (n == 2)
27     {
28         *r1 = a[0], *r2 = a[1];
29         return cabs(*r1 - *r2);
30     }
31     int m = n/2, i, j, k = 0;
32     pt r3, r4;
33     double p = closest_pair_r(a, b, m, r1, r2);
34     double d = closest_pair_r(a + m + 1, b + m + 1, n - m - 1, &r3, &r4);
35     if (d < p) p = d, *r1 = r3, *r2 = r4;
36     for (i = 0; i < n; i++) if (fabs(creal(a[i] - a[m])) < p) b[k++] = a[i];
37     qsort(b, k, sizeof(b[0]), cmpy);
38     for (i = 0; i < k; i++)
39         for (j = i + 1; cimag(b[j] - b[i]) < p && j < k; j++)
40             if (cabs(b[i] - b[j]) < p)
41                 p = cabs(b[i] - b[j]), *r1 = b[i], *r2 = b[j];
42     return p;
43 }
44
45 double closest_pair(pt* a, int n, pt* r1, pt* r2)
46 { // Skilar lengd milli nálægustu punkta í a: r1 og r2.
47     pt b[n];
48     qsort(a, n, sizeof(a[0]), cmpx);
49     return closest_pair_r(a, b, n, r1, r2);
50 }

```



- ▶ Hvert endurkvæmt kall er  $\mathcal{O}(\quad)$ .

- ▶ Hvert endurkvæmt kall er  $\mathcal{O}(n \log n)$ .

- ▶ Hvert endurkvæmt kall er  $\mathcal{O}(n \log n)$ .
- ▶ Svo tímaflækjan á þessari útfærslu er  $\mathcal{O}(\quad)$ , sem er bæting.

- ▶ Hvert endurkvæmt kall er  $\mathcal{O}(n \log n)$ .
- ▶ Svo tímaflækjan á þessari útfærslu er  $\mathcal{O}(n \log^2 n)$ , sem er bæting.

- ▶ Hvert endurkvæmt kall er  $\mathcal{O}(n \log n)$ .
- ▶ Svo tímaflækjan á þessari útfærslu er  $\mathcal{O}(n \log^2 n)$ , sem er bæting.
- ▶ Eina ástæðan fyrir því að hvert endurkvæmt kall sé  $\mathcal{O}(n \log n)$  er að við röðum í hvert skipti.

- ▶ Hvert endurkvæmt kall er  $\mathcal{O}(n \log n)$ .
- ▶ Svo tímaflækjan á þessari útfærslu er  $\mathcal{O}(n \log^2 n)$ , sem er bæting.
- ▶ Eina ástæðan fyrir því að hvert endurkvæmt kall sé  $\mathcal{O}(n \log n)$  er að við röðum í hvert skipti.
- ▶ Það er óþarfi því við erum alltaf að raða sömu punktunum aftur og aftur.

- ▶ Hvert endurkvæmt kall er  $\mathcal{O}(n \log n)$ .
- ▶ Svo tímaflækjan á þessari útfærslu er  $\mathcal{O}(n \log^2 n)$ , sem er bæting.
- ▶ Eina ástæðan fyrir því að hvert endurkvæmt kall sé  $\mathcal{O}(n \log n)$  er að við röðum í hvert skipti.
- ▶ Það er óþarfi því við erum alltaf að raða sömu punktunum aftur og aftur.
- ▶ Það eru fleiri en ein leið til að þurfa ekki að raða oft.

- ▶ Hvert endurkvæmt kall er  $\mathcal{O}(n \log n)$ .
- ▶ Svo tímaflækjan á þessari útfærslu er  $\mathcal{O}(n \log^2 n)$ , sem er bæting.
- ▶ Eina ástæðan fyrir því að hvert endurkvæmt kall sé  $\mathcal{O}(n \log n)$  er að við röðum í hvert skipti.
- ▶ Það er óþarfi því við erum alltaf að raða sömu punktunum aftur og aftur.
- ▶ Það eru fleiri en ein leið til að þurfa ekki að raða oft.
- ▶ Ein leið er að gera það sama og er gert í mergesort.



- ▶ Hvert endurkvæmt kall er  $\mathcal{O}(n \log n)$ .
- ▶ Svo tímaflækjan á þessari útfærslu er  $\mathcal{O}(n \log^2 n)$ , sem er bæting.
- ▶ Eina ástæðan fyrir því að hvert endurkvæmt kall sé  $\mathcal{O}(n \log n)$  er að við röðum í hvert skipti.
- ▶ Það er óþarfi því við erum alltaf að raða sömu punktunum aftur og aftur.
- ▶ Það eru fleiri en ein leið til að þurfa ekki að raða oft.
- ▶ Ein leið er að gera það sama og er gert í mergesort.
- ▶ Þá erum við í raun að raða, en við gerum það í línulegum tíma.

```

15 void merge(pt* a, pt *b, int m, int r)
16 { // Úr D&D glærunum.
17   int i = 0, j = m, c = 0;
18   while (i < m && j < r) b[c++] = a[cimag(a[j]) < cimag(a[i]) ? j++ : i++];
19   while (i < m || j < r) b[c++] = a[j < r ? j++ : i++];
20   for (i = 0; i < r; i++) a[i] = b[i];
21 }
22
23 double closest_pair_r(pt *a, pt *b, pt *c, int n, pt *r1, pt *r2)
24 { // Hjálparfall
25   if (n < 2) { b[0] = a[0]; return 1e16; }
26   if (n == 2)
27   {
28     *r1 = b[0] = a[0], *r2 = b[1] = a[1], merge(b, c, 1, 2);
29     return cnorm(*r1 - *r2);
30   }
31   int m = n/2, i, j, k = 0;
32   pt r3, r4;
33   double p = closest_pair_r(a, b, c, m, r1, r2), d;
34   b[m] = a[m];
35   d = closest_pair_r(a + m + 1, b + m + 1, c + m + 1, n - m - 1, &r3, &r4);
36   if (d < p) p = d, *r1 = r3, *r2 = r4;
37   merge(b, c, m, m + 1), merge(b, c, m + 1, n);
38   for (i = 0; i < n; i++) if (fabs(creal(b[i] - a[m])) < p) c[k++] = b[i];
39   for (i = 0; i < k; i++)
40     for (j = i + 1; cimag(c[j] - c[i]) < p && j < k; j++)
41       if (cabs(c[i] - c[j]) < p)
42         p = cabs(c[i] - c[j]), *r1 = c[i], *r2 = c[j];
43   return p;
44 }
45
46 double closest_pair(pt *a, int n, pt *r1, pt *r2)
47 { // Skilar lengd milli nálægustu punkta í a: r1 og r2.
48   pt b[n], c[n];
49   qsort(a, n, sizeof(a[0]), cmpx);
50   return closest_pair_r(a, b, c, n, r1, r2);
51 }

```

- ▶ Hvert endurkvæmt kall er nú  $\mathcal{O}(\quad)$ .

- ▶ Hvert endurkvæmt kall er nú  $\mathcal{O}(n)$ .

- ▶ Hvert endurkvæmt kall er nú  $\mathcal{O}(n)$ .
- ▶ Svo tímaflækjan á þessari útfærslu er  $\mathcal{O}(\quad)$ .

- ▶ Hvert endurkvæmt kall er nú  $\mathcal{O}(n)$ .
- ▶ Svo tímaflækjan á þessari útfærslu er  $\mathcal{O}(n \log n)$ .

- ▶ Hvert endurkvæmt kall er nú  $\mathcal{O}(n)$ .
- ▶ Svo tímaflækjan á þessari útfærslu er  $\mathcal{O}(n \log n)$ .
- ▶ Þetta má síðan bæta með slembnum reikniritum, en verður annars ekki betra.

