

Lausnir á dæmum tengd viku tólf

Bergur Snorrason

29. mars 2022

- ▶ Ég mun leysa eftirfarandi dæmi:

- ▶ Ég mun leysa eftirfarandi dæmi:
 - ▶ *Maximum Number of Colinear Points,*

- ▶ Ég mun leysa eftirfarandi dæmi:
 - ▶ *Maximum Number of Colinear Points,*
 - ▶

Maximum Number of Colinear Points

- ▶ Gefnir eru n punktar í plani.

Maximum Number of Colinear Points

- ▶ Gefnir eru n punktar í plani.
- ▶ Þú vilt velja hlutmengi af þessum punktum þannig að allir punktarnir í hlutmenginu liggi á sömu línunni.

Maximum Number of Colinear Points

- ▶ Gefnir eru n punktar í plani.
- ▶ Þú vilt velja hlutmengi af þessum punktum þannig að allir punktarnir í hlutmenginu liggi á sömu línunni.
- ▶ Hvert er stærðin á stærstu hlutmengjunum sem þú getur valið.

- ▶ Hvert par af punktum skilgreinir línu.

- ▶ Hvert par af punktum skilgreinir línu.
- ▶ Við getum því fyrir sérhvert par af punktum ítrað í gegnum alla punktanna og séð hvað margir liggja á línunni.

- ▶ Hvert par af punktum skilgreinir línu.
- ▶ Við getum því fyrir sérhvert par af punktum ítrað í gegnum alla punktanna og séð hvað margir liggja á línunni.
- ▶ Þessi lausn er $\mathcal{O}(n^3)$.

- ▶ Hvert par af punktum skilgreinir línu.
- ▶ Við getum því fyrir sérhvert par af punktum ítrað í gegnum alla punktanna og séð hvað margir liggja á línunni.
- ▶ Þessi lausn er $\mathcal{O}(n^3)$.
- ▶ Reynum að bæta þetta.

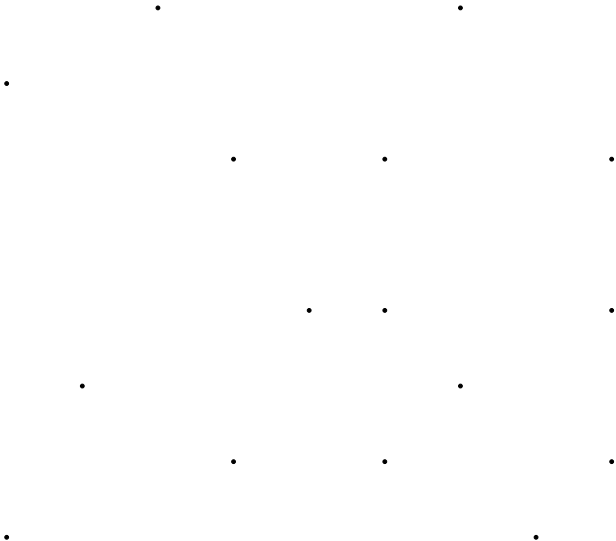
- ▶ Veljum einn punkt sem vendipunkt og röðum öðrum punktum miðað við stenfhornið sem þeir mynda við vendipunktinn.

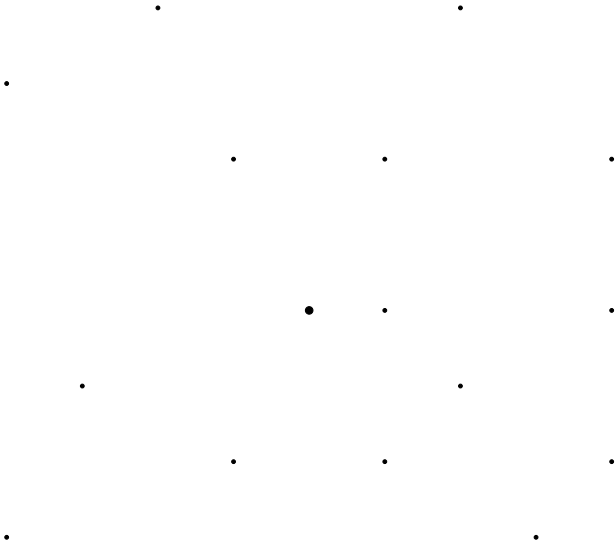
- ▶ Veljum einn punkt sem vendipunkt og röðum öðrum punktum miðað við stenfhornið sem þeir mynda við vendipunktinn.
- ▶ Þá eru allir punktar sem liggja á sama geisla sem byrjar í vendipunktinum aðlægir.

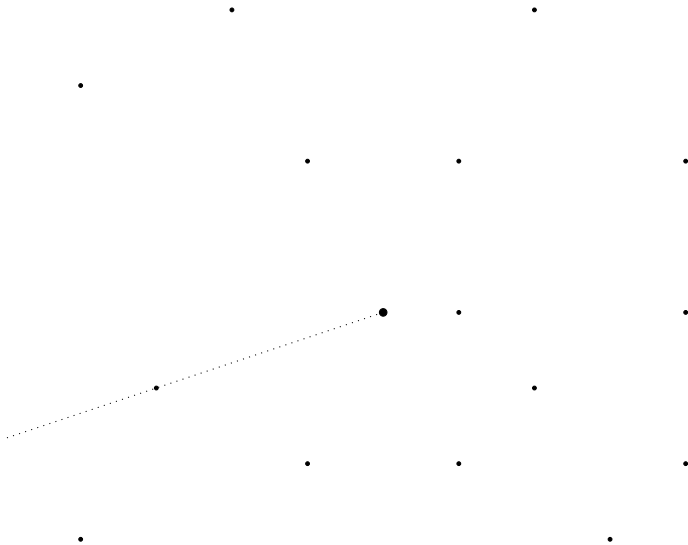
- ▶ Veljum einn punkt sem vendipunkt og röðum öðrum punktum miðað við stenhornið sem þeir mynda við vendipunktinn.
- ▶ Þá eru allir punktar sem liggja á sama geisla sem byrjar í vendipunktinum aðlægir.
- ▶ Við getum því gengið einu sinni í gegnum punktanna og fundið besta svarið að því gefnum að vendipunkturinn liggi á línunni.

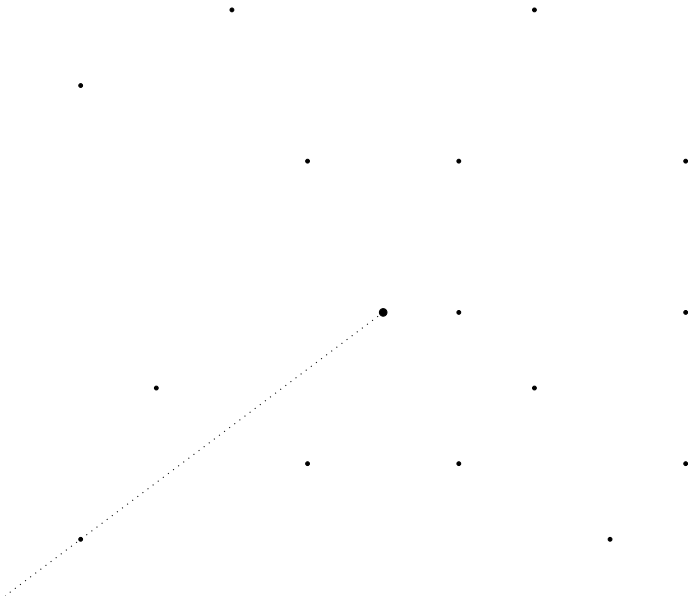
- ▶ Veljum einn punkt sem vendipunkt og röðum öðrum punktum miðað við stefnhornið sem þeir mynda við vendipunktinn.
- ▶ Þá eru allir punktar sem liggja á sama geisla sem byrjar í vendipunktinum aðlægir.
- ▶ Við getum því gengið einu sinni í gegnum punktanna og fundið besta svarið að því gefnum að vendipunkturinn liggi á línunni.
- ▶ Við getum endurtekið þetta n sinnum, þannig að hver punktur fær að vera vendipunktur.

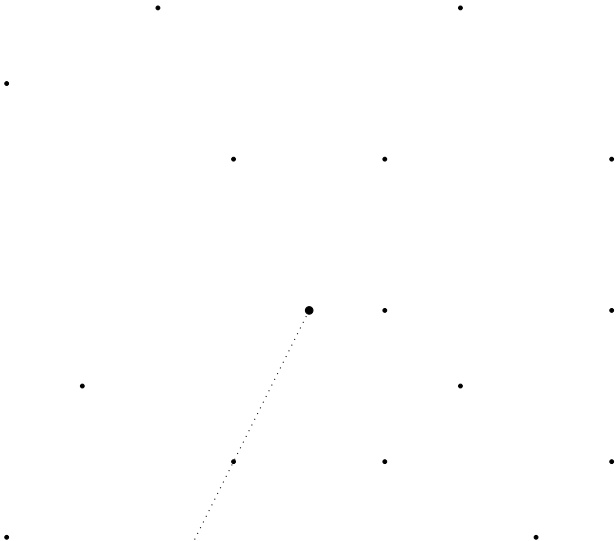
- ▶ Veljum einn punkt sem vendipunkt og röðum öðrum punktum miðað við stenfhornið sem þeir mynda við vendipunktinn.
- ▶ Þá eru allir punktar sem liggja á sama geisla sem byrjar í vendipunktinum aðlægir.
- ▶ Við getum því gengið einu sinni í gegnum punktanna og fundið besta svarið að því gefnum að vendipunkturinn liggi á línunni.
- ▶ Við getum endurtekið þetta n sinnum, þannig að hver punktur fær að vera vendipunktur.
- ▶ Tökum sýnidæmi.

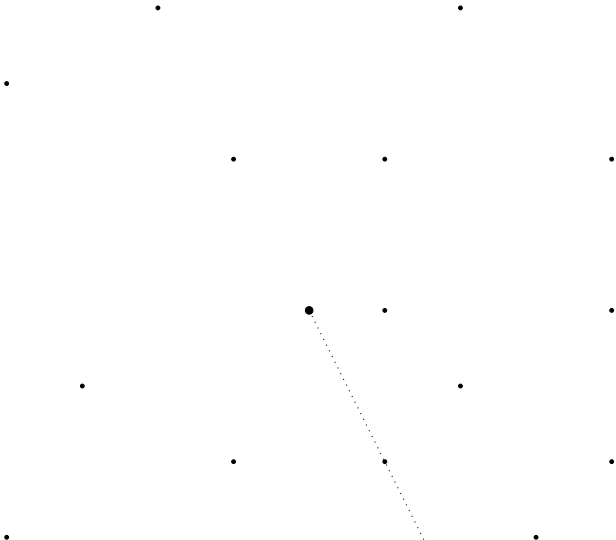


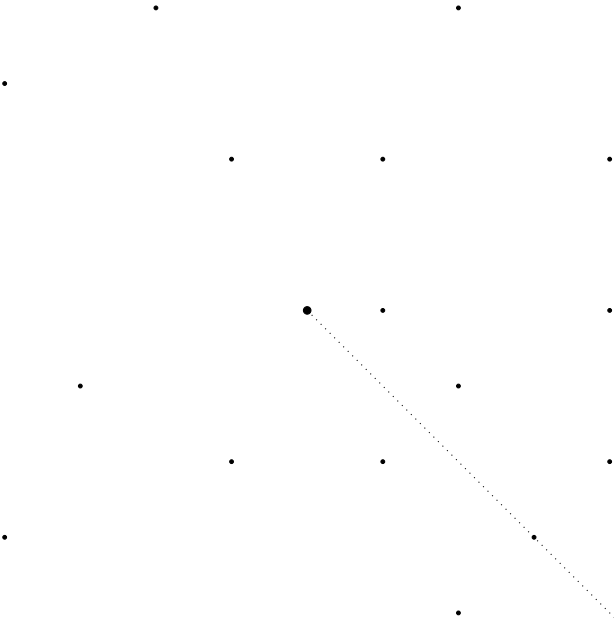


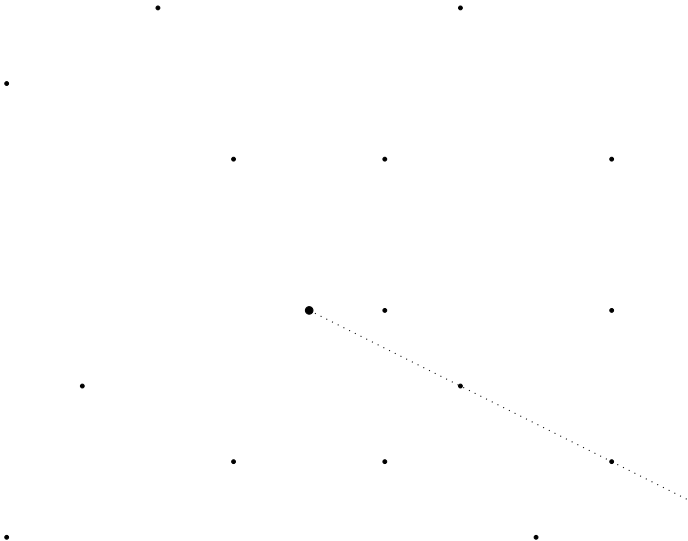




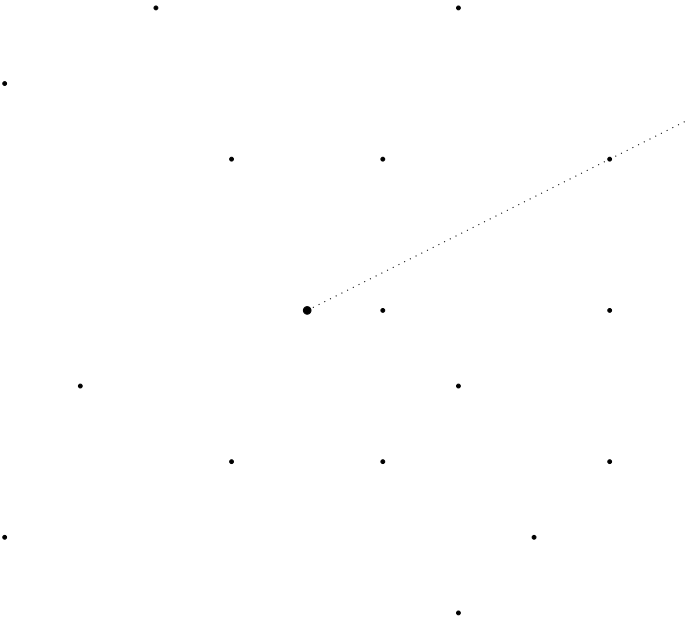


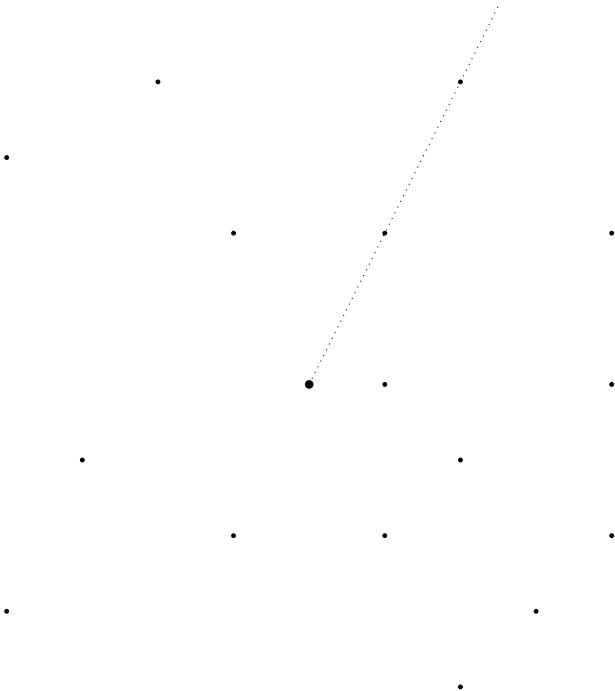


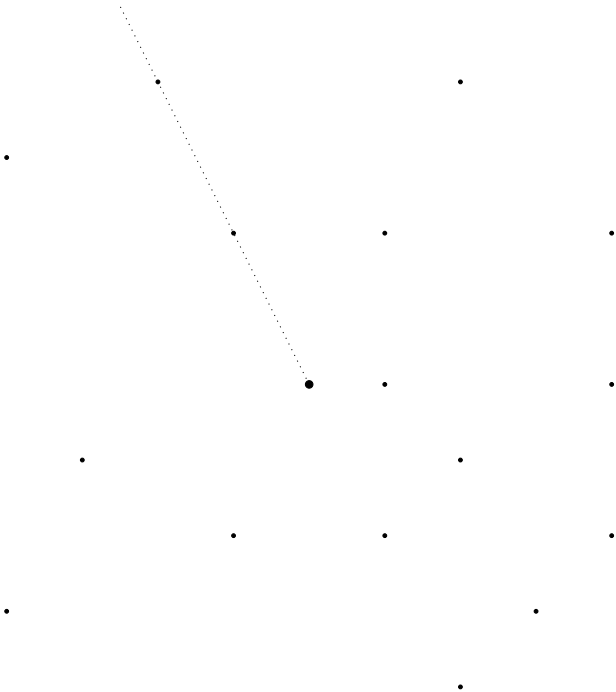


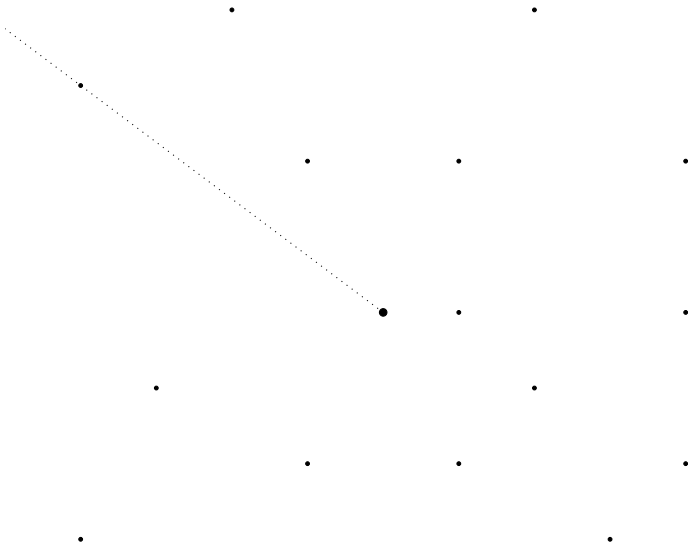


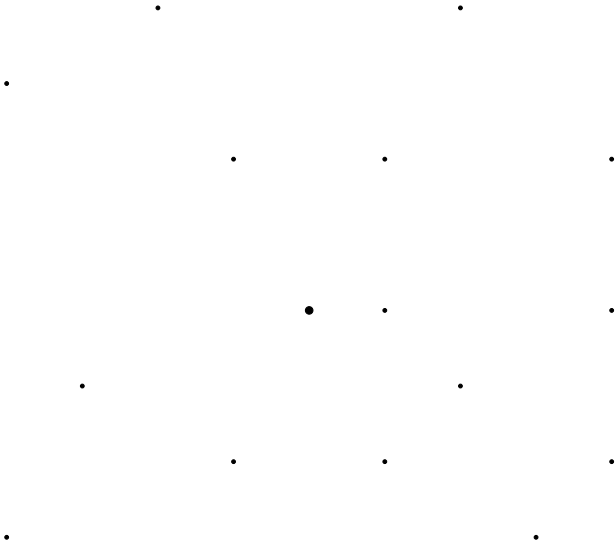


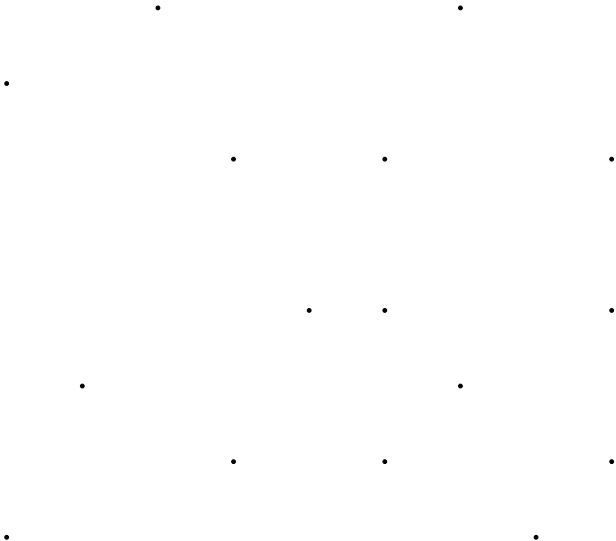


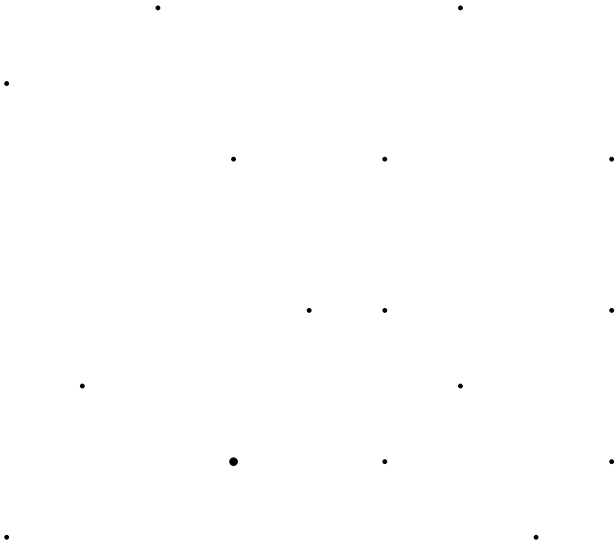


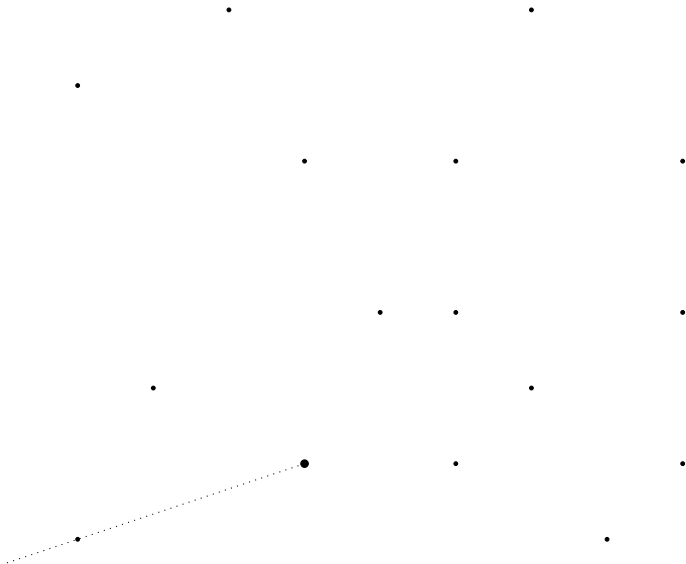


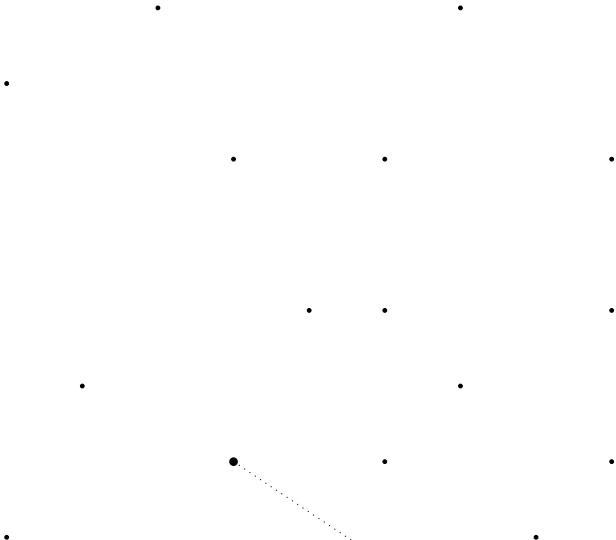


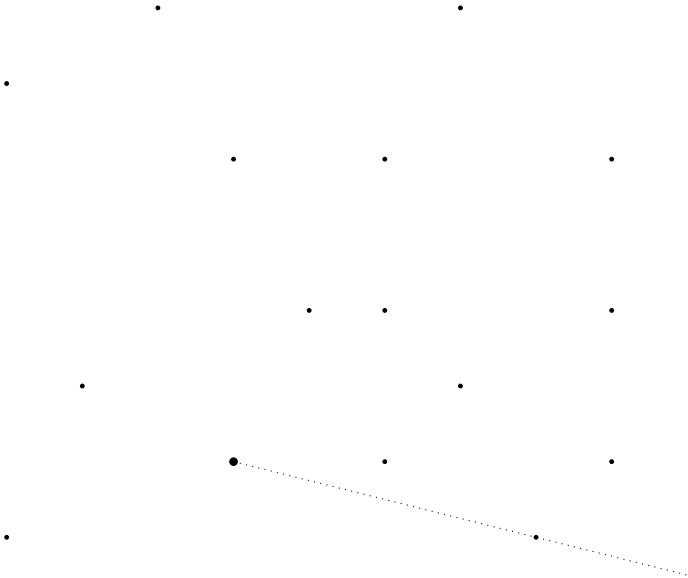




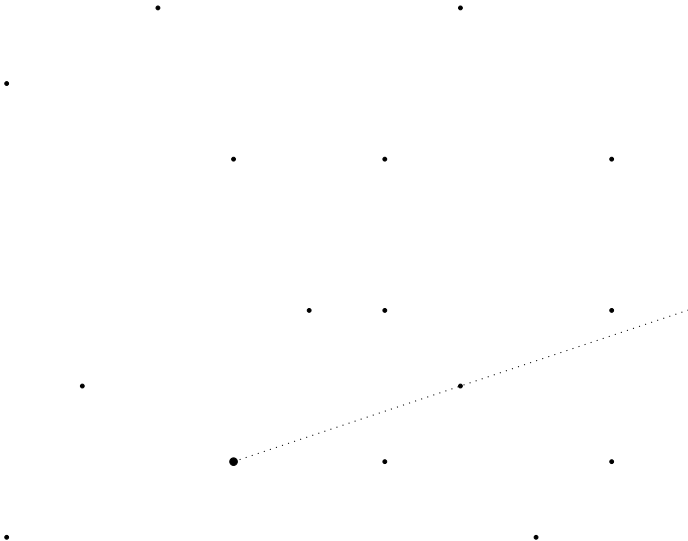


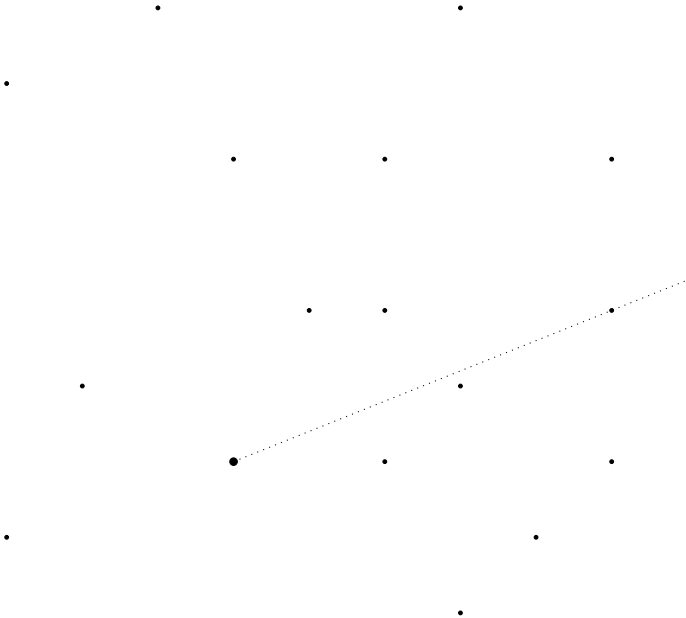


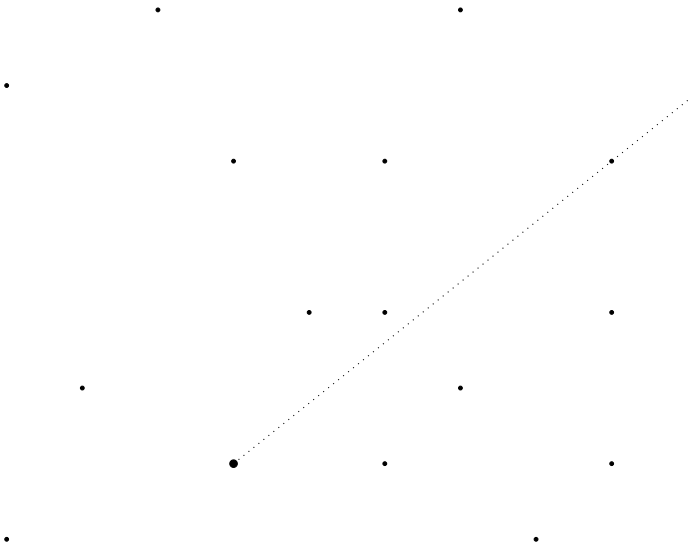


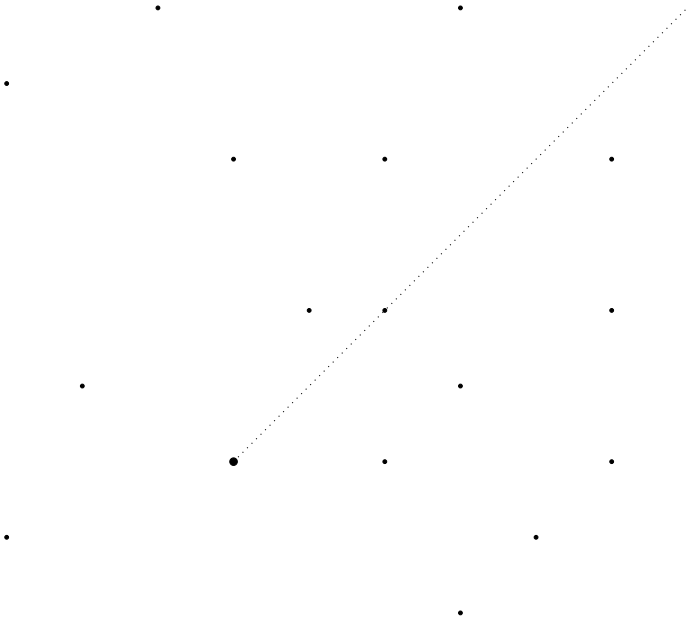


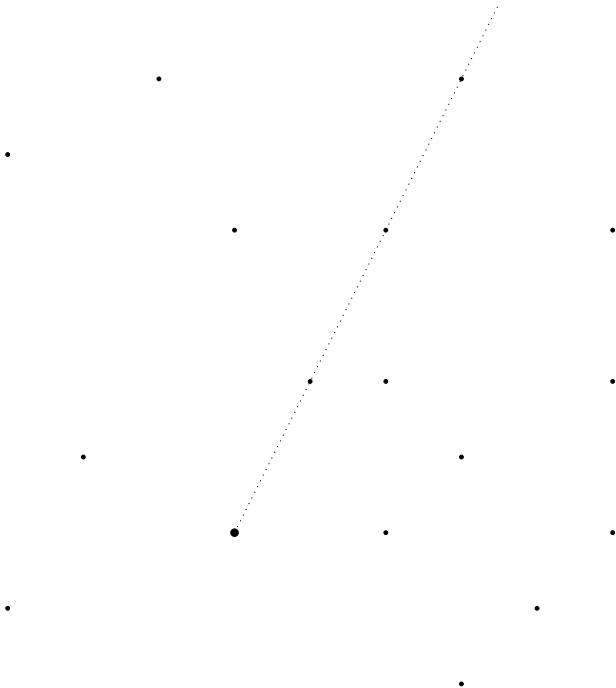


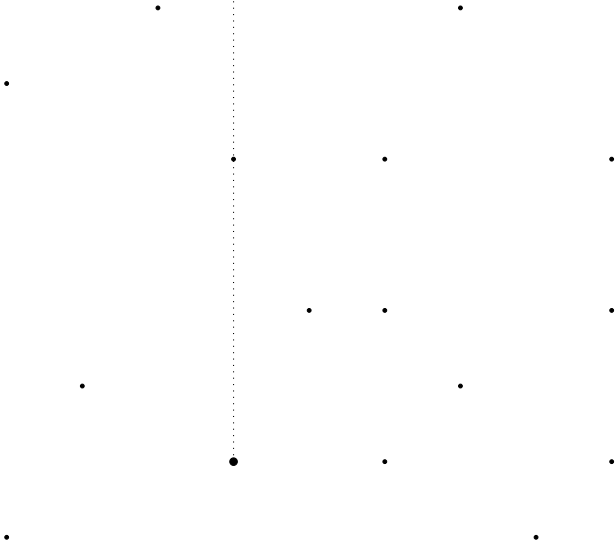


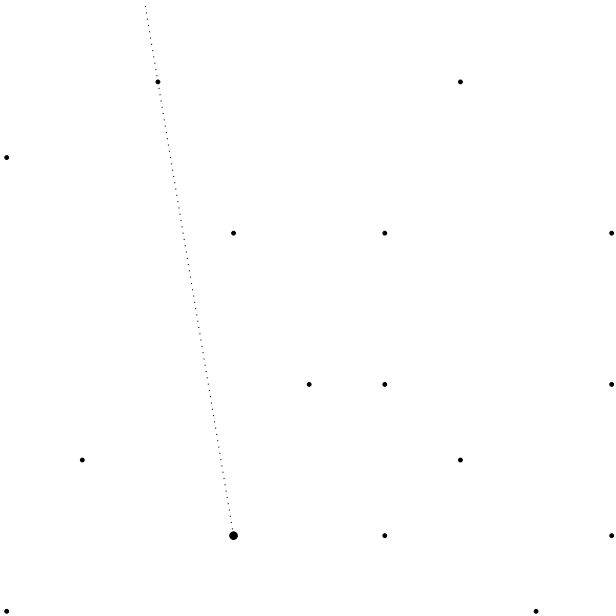


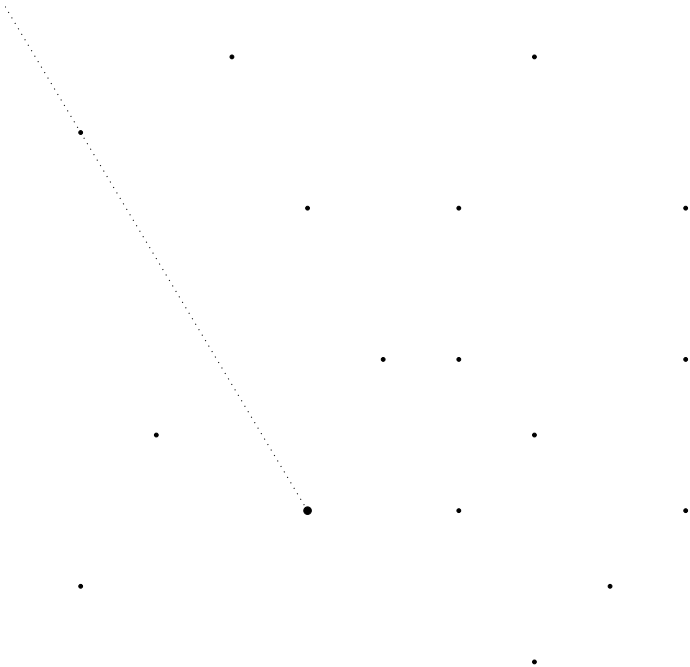


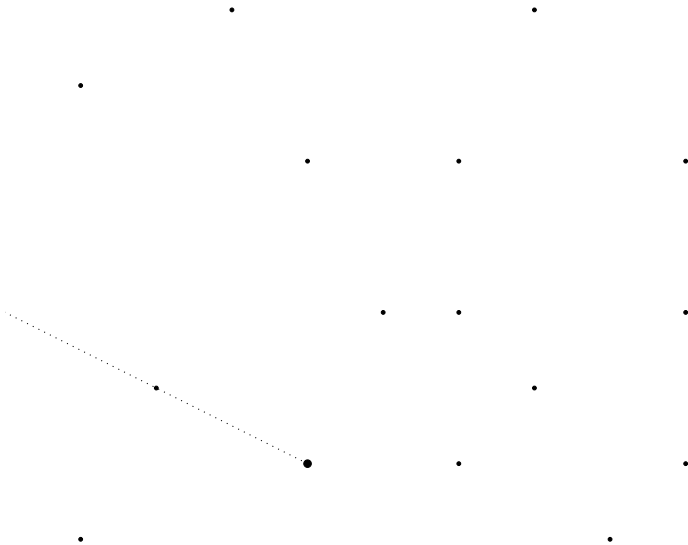


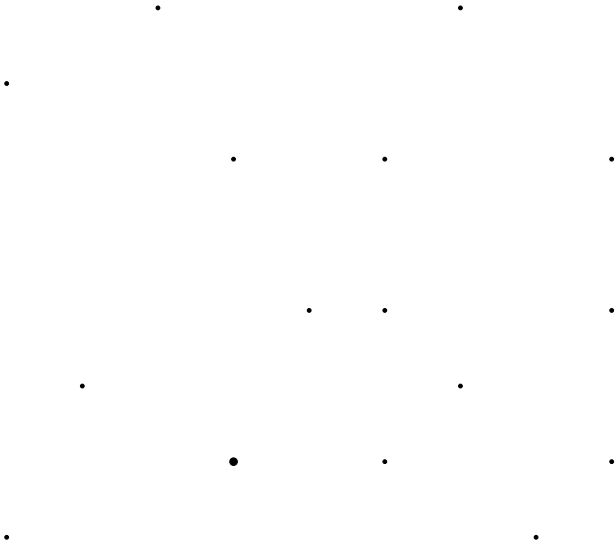


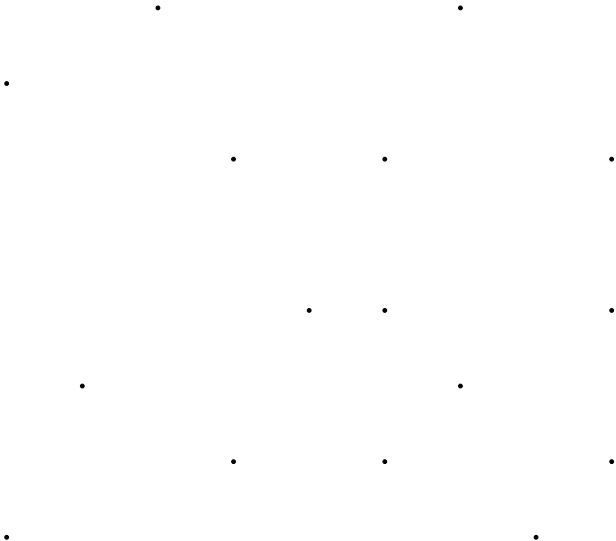


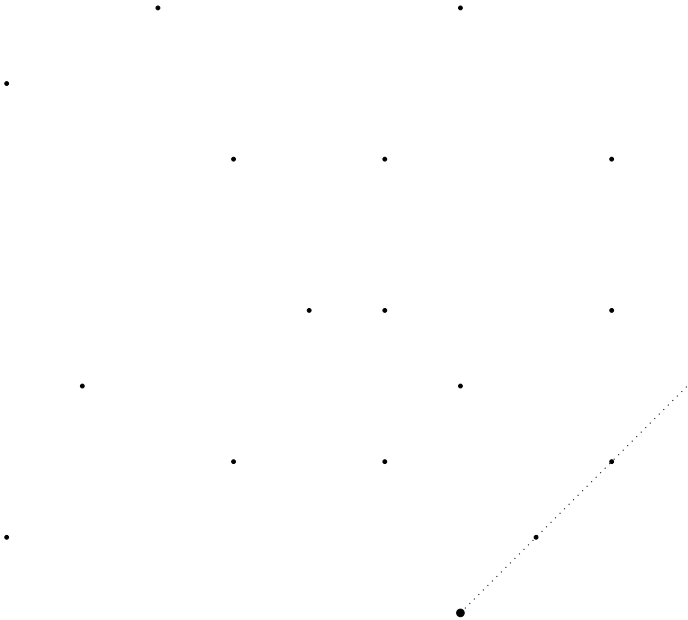


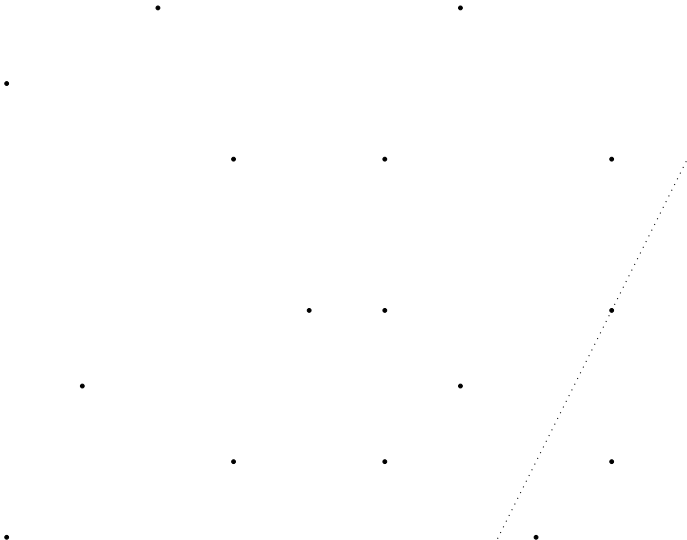


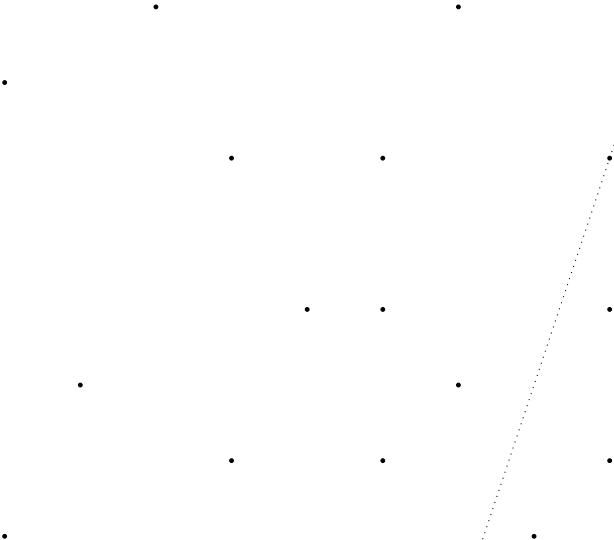


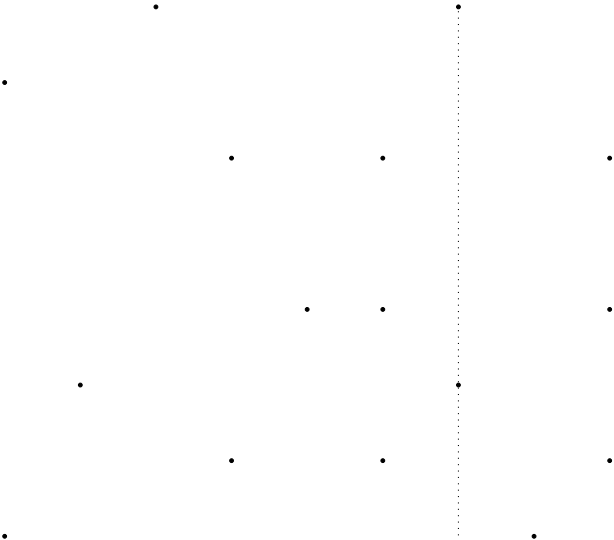


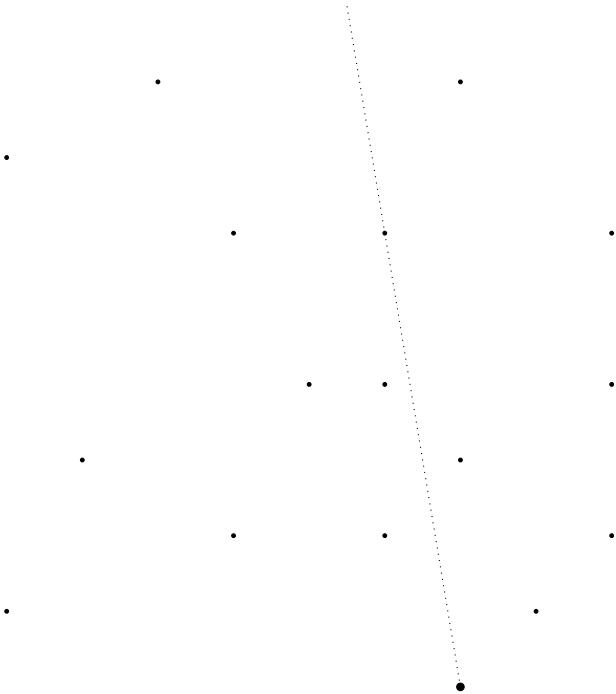


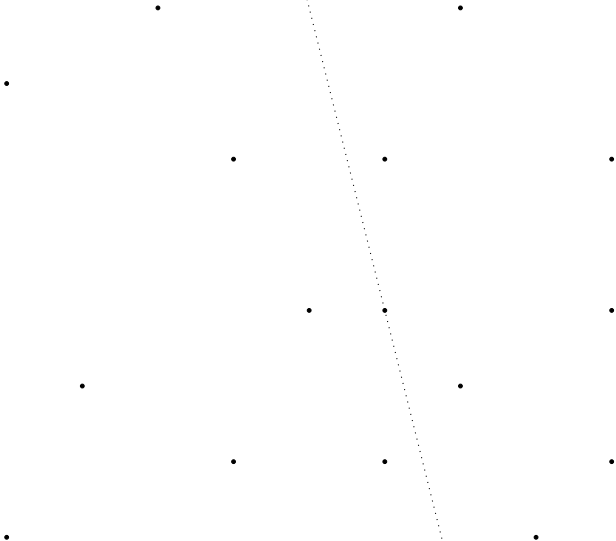


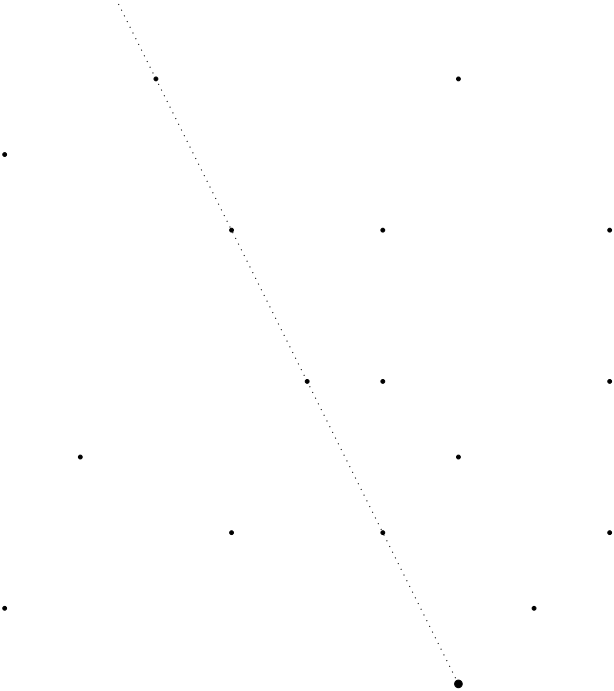


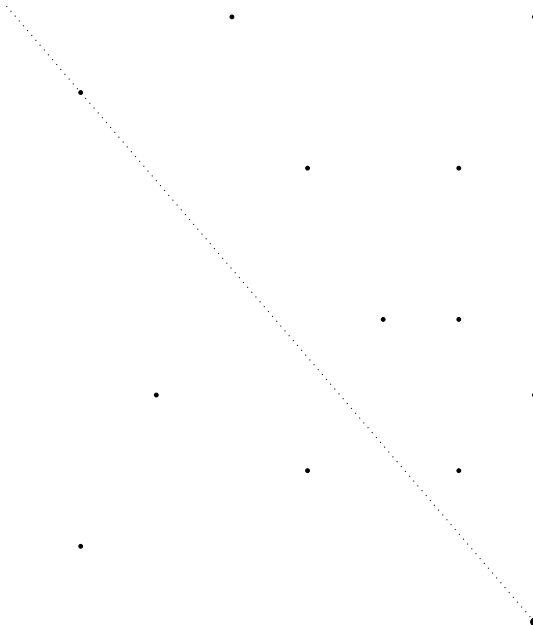


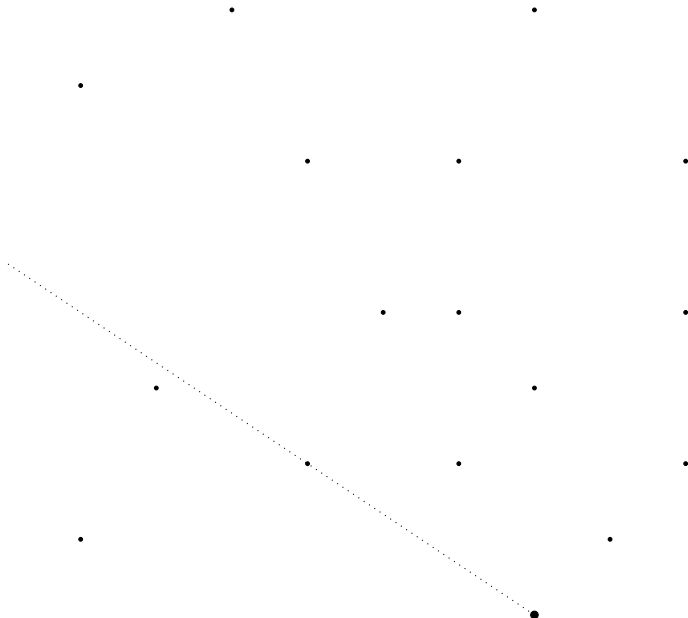


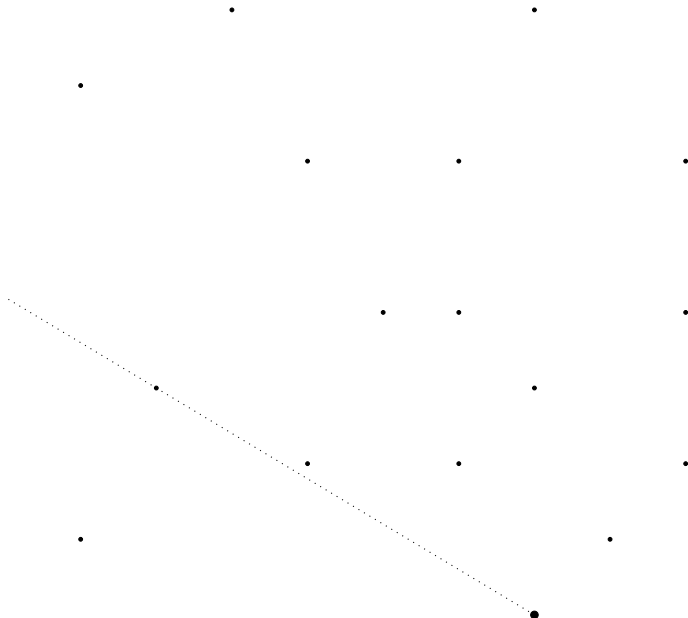


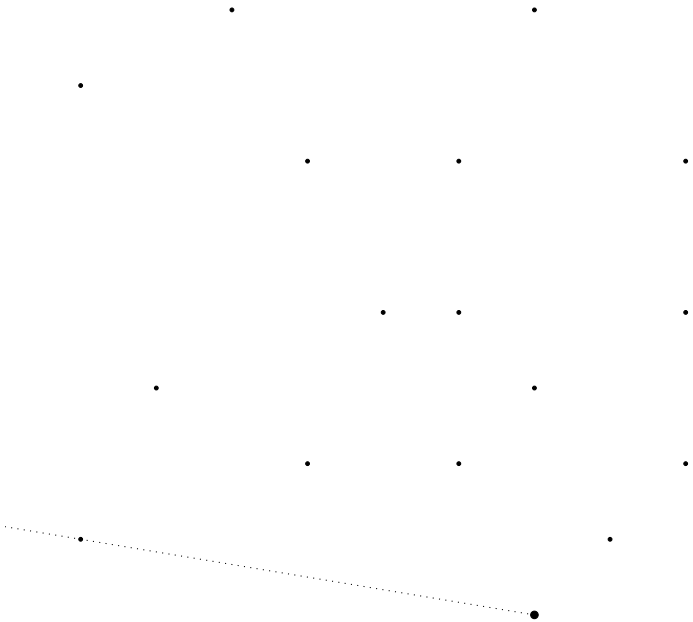


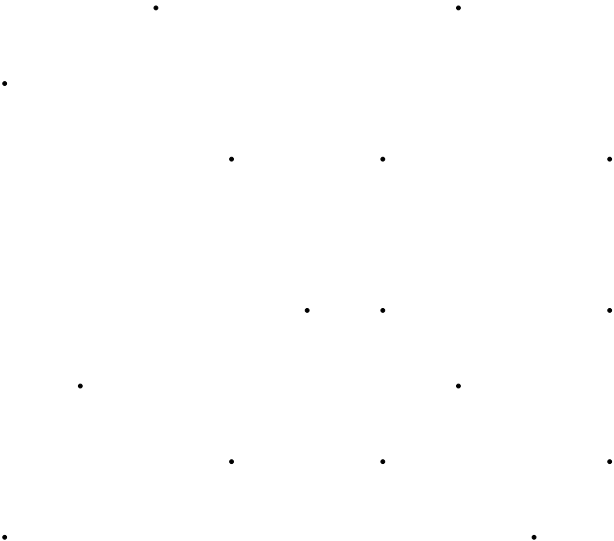


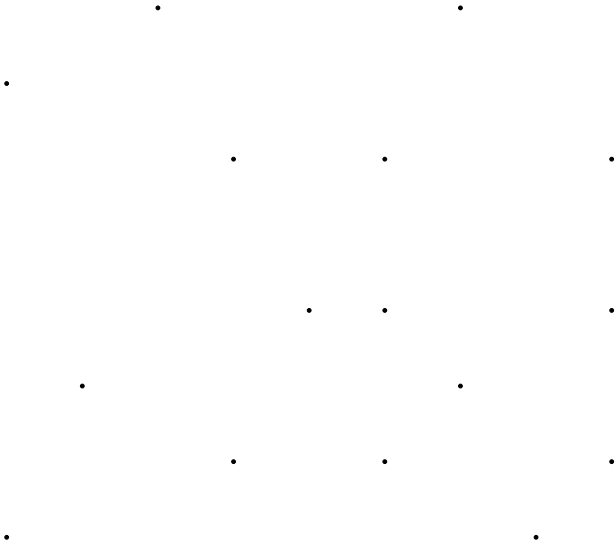












- ▶ Við köllum þessa aðferð *sópin* (e. *sweep line*).

- ▶ Við köllum þessa aðferð *sópin* (e. *sweep line*).
- ▶ Dæmi sem má leysa með sópnum:

- ▶ Við köllum þessa aðferð *sópin* (e. *sweep line*).
- ▶ Dæmi sem má leysa með sópnum:
 - ▶ Finna nálægustu tvo punkta í punktasafni.

- ▶ Við köllum þessa aðferð *sóþinn* (e. *sweep line*).
- ▶ Dæmi sem má leysa með sóþnum:
 - ▶ Finna nálægustu tvo punkta í punktasafni.
 - ▶ Finna Delaunay net punktasafns (reiknirit Fortunes).

- ▶ Við köllum þessa aðferð *sóþinn* (e. *sweep line*).
- ▶ Dæmi sem má leysa með sóþnum:
 - ▶ Finna nálægustu tvo punkta í punktasafni.
 - ▶ Finna Delaunay net punktasafns (reiknirit Fortunes).
 - ▶ Athuga hvort safn línustrika skerist (reiknirit Shamos og Hoey).

- ▶ Við köllum þessa aðferð *sópin* (e. *sweep line*).
- ▶ Dæmi sem má leysa með sópnum:
 - ▶ Finna nálægustu tvo punkta í punktastafi.
 - ▶ Finna Delaunay net punktastafs (reiknirit Fortunes).
 - ▶ Athuga hvort safn línustrika skerist (reiknirit Shamos og Hoey).
- ▶ Þessi dæmi eiga það öll sameiginlegt að við drögum beina línu (yfirleitt samsíða y -ásnum) í gegnum punktastafið okkar.

- ▶ Við köllum þessa aðferð *sópin* (e. *sweep line*).
- ▶ Dæmi sem má leysa með sópnum:
 - ▶ Finna nálægustu tvo punkta í punktastafi.
 - ▶ Finna Delaunay net punktastafs (reiknirit Fortunes).
 - ▶ Athuga hvort safn línustrika skerist (reiknirit Shamos og Hoey).
- ▶ Þessi dæmi eiga það öll sameiginlegt að við drögum beina línu (yfirléitt samsíða y -ásnum) í gegnum punktastafið okkar.
- ▶ Dæmið sem við erum að skoða snýr línu með fasta miðju, svo kallaður *snúningssópur*.

- ▶ Við köllum þessa aðferð *sópin* (e. *sweep line*).
- ▶ Dæmi sem má leysa með sópnum:
 - ▶ Finna nálægustu tvo punkta í punktastafi.
 - ▶ Finna Delaunay net punktastafs (reiknirit Fortunes).
 - ▶ Athuga hvort safn línustrika skerist (reiknirit Shamos og Hoey).
- ▶ Þessi dæmi eiga það öll sameiginlegt að við drögum beina línu (yfirléitt samsíða y -ásnum) í gegnum punktastafið okkar.
- ▶ Dæmið sem við erum að skoða snýr línu með fasta miðju, svo kallaður *snúningssópur*.
- ▶ Snúningssópurinn er algengari í dæmum.

- ▶ Nýtum okkur þetta tækifæri og skoðum hvernig við getum útfært rúmfræði í heiltölum.

- ▶ Nýtum okkur þetta tækifæri og skoðum hvernig við getum útfært rúmfræði í heiltölum.
- ▶ Heilsti kosturinn við þessa aðferð er að við getum leyst dæmið án fleytitöluskekkju.

- ▶ Nýtum okkur þetta tækifæri og skoðum hvernig við getum útfært rúmfræði í heiltölum.
- ▶ Heilsti kosturinn við þessa aðferð er að við getum leyst dæmið án fleytitöluskekkju.
- ▶ Við geymum þá punkt sem tvennd af heiltölum.

- ▶ Nýtum okkur þetta tækifæri og skoðum hvernig við getum útfært rúmfræði í heiltölum.
- ▶ Heilsti kosturinn við þessa aðferð er að við getum leyst dæmið án fleytitöluskekkju.
- ▶ Við geymum þá punkt sem tvennd af heiltölum.
- ▶ Við þurfum að geta raðað punktunum eftir stefnuhorni.

- ▶ Nýtum okkur þetta tækifæri og skoðum hvernig við getum útfært rúmfræði í heiltölum.
- ▶ Heilsti kosturinn við þessa aðferð er að við getum leyst dæmið án fleytitöluskekkju.
- ▶ Við geymum þá punkt sem tvennd af heiltölum.
- ▶ Við þurfum að geta raðað punktunum eftir stefnuhorni.
- ▶ Við getum gert þetta án þess að reikna stefnu hornið.

- ▶ Nýtum okkur þetta tækifæri og skoðum hvernig við getum útfært rúmfræði í heiltölum.
- ▶ Heilsti kosturinn við þessa aðferð er að við getum leyst dæmið án fleytitöluskekkju.
- ▶ Við geymum þá punkt sem tvennd af heiltölum.
- ▶ Við þurfum að geta raðað punktunum eftir stefnuhorni.
- ▶ Við getum gert þetta án þess að reikna stefnu hornið.
- ▶ Köllum vendipunktinn P og punktanna sem við viljum bera saman X og Y .

- ▶ Nýtum okkur þetta tækifæri og skoðum hvernig við getum útfært rúmfræði í heiltölum.
- ▶ Heilsti kosturinn við þessa aðferð er að við getum leyst dæmið án fleytitöluskekkju.
- ▶ Við geymum þá punkt sem tvennd af heiltölum.
- ▶ Við þurfum að geta raðað punktunum eftir stefnuhorni.
- ▶ Við getum gert þetta án þess að reikna stefnu hornið.
- ▶ Köllum vendipunktinn P og punktanna sem við viljum bera saman X og Y .
- ▶ Berum fyrst saman X og Y eftir því hvort þeir séu fyrir ofan eða neðan vendipunktinn.

- ▶ Nýtum okkur þetta tækifæri og skoðum hvernig við getum útfært rúmfræði í heiltölum.
- ▶ Heilsti kosturinn við þessa aðferð er að við getum leyst dæmið án fleytitöluskekkju.
- ▶ Við geymum þá punkt sem tvennd af heiltölum.
- ▶ Við þurfum að geta raðað punktunum eftir stefnuhorni.
- ▶ Við getum gert þetta án þess að reikna stefnu hornið.
- ▶ Köllum vendipunktinn P og punktanna sem við viljum bera saman X og Y .
- ▶ Berum fyrst saman X og Y eftir því hvort þeir séu fyrir ofan eða neðan vendipunktinn.
- ▶ Punktar fyrir ofan fá minni forgang.

- Til að bera saman punktana X og Y ef þeir eru báðir fyrir ofan eða neðan vendipunktinn þá athugum við í hvaða átt við beygjum þegar við löbbum frá X til Y í gegnum punktinn P .

- ▶ Til að bera saman punktana X og Y ef þeir eru báðir fyrir ofan eða neðan vendipunktinn þá athugum við í hvaða átt við beygjum þegar við löbbum frá X til Y í gegnum punktinn P .
- ▶ Punkturinn X hefur þá meiri forgang ef beygjan er til hægri.

```

6 typedef struct { ll x, y; } pt;
7 pt topt(ll x, ll y) { pt r = {x, y}; return r; }
8 ll ccw(pt a, pt b, pt c)
9 {
10     ll r = a.x*(b.y - c.y) + b.x*(c.y - a.y) + c.x*(a.y - b.y);
11     if (r == 0) return r;
12     return r < 0 ? 1 : -1;
13 }
14
15 ll above(pt a)
16 {
17     if (a.x == 0 && a.y == 0) return 2;
18     return a.y < 0 ? 0 : 1;
19 }
20
21 int cmp(const void *p1, const void *p2)
22 {
23     pt a = *(pt*)p1, b = *(pt*)p2;
24     ll c = above(a), d = above(b);
25     return c != d ? c - d : ccw(a, topt(0, 0), b);
26 }

```



```

28 int main()
29 {
30     ll i, j, k, n, r;
31     scanf("%lld", &n);
32     while (n)
33     {
34         pt a[n], b[n];
35         for (i = 0; i < n; i++) scanf("%lld%lld", &a[i].x, &a[i].y);
36         if (n == 1) { printf("1\n"); scanf("%lld", &n); continue; }
37         for (r = 2, i = 0; i < n; i++)
38         {
39             for (j = 0; j < n; j++)
40                 b[j].x = a[j].x - a[i].x, b[j].y = a[j].y - a[i].y;
41             qsort(b, n, sizeof *b, cmp);
42             for (k = 2, j = 1; j < n - 1; j++)
43                 (ccw(b[j], b[j - 1], b[n - 1]) != 0)
44                     ? (k = 2) : (r = max(r, ++k));
45         }
46         printf("%d\n", r);
47         scanf("%lld", &n);
48     }
49     return 0;
50 }

```

- ▶ Við framkvæmum n sópa, einn fyrir hvern punkt.

- ▶ Við framkvæmum n sópa, einn fyrir hvern punkt.
- ▶ Í hverjum sóp röðum við punktasetninu, og löbbu svo í gegnum það einu sinni.

- ▶ Við framkvæmum n sópa, einn fyrir hvern punkt.
- ▶ Í hverjum sóp röðum við punktasetninu, og löbbu svo í gegnum það einu sinni.
- ▶ Svo tímaflækjan á hverjum sóp er $\mathcal{O}(\quad)$.

- ▶ Við framkvæmum n sópa, einn fyrir hvern punkt.
- ▶ Í hverjum sóp röðum við punktastafninu, og löbbu svo í gegnum það einu sinni.
- ▶ Svo tímaflækjan á hverjum sóp er $\mathcal{O}(n \log n)$.

- ▶ Við framkvæmum n sópa, einn fyrir hvern punkt.
- ▶ Í hverjum sóp röðum við punktasafninu, og löbbu svo í gegnum það einu sinni.
- ▶ Svo tímaflækjan á hverjum sóp er $\mathcal{O}(n \log n)$.
- ▶ Svo tímaflækjan í heildina er $\mathcal{O}(\quad)$.

- ▶ Við framkvæmum n sópa, einn fyrir hvern punkt.
- ▶ Í hverjum sóp röðum við punktasafninu, og löbbu svo í gegnum það einu sinni.
- ▶ Svo tímaflækjan á hverjum sóp er $\mathcal{O}(n \log n)$.
- ▶ Svo tímaflækjan í heildina er $\mathcal{O}(n^2 \log n)$.

- ▶ Skoďum aďra aďferď.

- ▶ Skoðum aðra aðferð.
- ▶ Látum L_{jk} tákna línuna gegnum j -ta og k -ta punktinn, $j < k$.

- ▶ Skoðum aðra aðferð.
- ▶ Látum L_{jk} tákna línuna gegnum j -ta og k -ta punktinn, $j < k$.
- ▶ Látum svo S_{jk} , $j < k$, tákna fjöld tvennda (j_0, k_0) þannig að $j_0 < k_0$ og $L_{j_0 k_0} = L_{jk}$.

- ▶ Skoðum aðra aðferð.
- ▶ Látum L_{jk} tákna línuna gegnum j -ta og k -ta punktinn, $j < k$.
- ▶ Látum svo S_{jk} , $j < k$, tákna fjöld tvennda (j_0, k_0) þannig að $j_0 < k_0$ og $L_{j_0 k_0} = L_{jk}$.
- ▶ Með öðrum orðum táknar S_{jk} hversu mörg eintök eru af línunni L_{jk} .

- ▶ Skoðum aðra aðferð.
- ▶ Látum L_{jk} tákna línuna gegnum j -ta og k -ta punktinn, $j < k$.
- ▶ Látum svo S_{jk} , $j < k$, tákna fjöld tvennda (j_0, k_0) þannig að $j_0 < k_0$ og $L_{j_0 k_0} = L_{jk}$.
- ▶ Með öðrum orðum táknar S_{jk} hversu mörg eintök eru af línunni L_{jk} .
- ▶ Línan sem flestir punktar liggja er einnig línan sem kemur oftast fyrir.

- ▶ Skoðum aðra aðferð.
- ▶ Látum L_{jk} tákna línuna gegnum j -ta og k -ta punktinn, $j < k$.
- ▶ Látum svo S_{jk} , $j < k$, tákna fjöld tvennda (j_0, k_0) þannig að $j_0 < k_0$ og $L_{j_0 k_0} = L_{jk}$.
- ▶ Með öðrum orðum táknar S_{jk} hversu mörg eintök eru af línunni L_{jk} .
- ▶ Línan sem flestir punktar liggja er einnig línan sem kemur oftast fyrir.
- ▶ Við getum fundið þá línu með því að raða línunum of telja.

- ▶ Tökum eftir að ef k punktar liggja á línunni L_{jk} þá er $S_{jk} = k(k - 1)/2$.

- ▶ Tökum eftir að ef k punktar liggja á línunni L_{jk} þá er $S_{jk} = k(k - 1)/2$.
- ▶ Svo ef við leysum þessa jöfnu fæst að svarið er $(1 + \sqrt{1 - 8M})/2$ þar sem M er stærsta gildið á S_{jk} sem við getum fengið.

```

7 typedef struct { double x, s; } lina;
8 lina tolina(double x, double s) { lina r = {x, s}; return r; }
9 int cmp(const void *p1, const void *p2)
10 {
11     lina a = *(lina*)p1, b = *(lina*)p2;
12     if (fabs(a.s - b.s) < EPS && fabs(a.x - b.x) < EPS) return 0;
13     if (fabs(a.s - b.s) < EPS) return a.x < b.x ? -1 : 1;
14     return a.s < b.s ? -1 : 1;
15 }

```



```

17 int main()
18 {
19     ll i, j, n;
20     scanf("%lld", &n);
21     while (n)
22     {
23         double x[n], y[n];
24         ll k = 0, r = 0;
25         lina a[n*n];
26         for (i = 0; i < n; i++) scanf("%lf%lf", &x[i], &y[i]);
27         if (n == 1) { printf("1\n"); scanf("%lld", &n); continue; }
28         for (i = 0; i < n; i++) for (j = 0; j < i; j++)
29         {
30             if (fabs(x[i] - x[j]) < EPS) a[k++] = tolina(x[i], 1e18);
31             else
32             {
33                 double s = (y[i] - y[j]) / (x[i] - x[j]);
34                 a[k++] = tolina(y[i] - s*x[i], s);
35             }
36         }
37         qsort(a, k, sizeof *a, cmp);
38         i = 0;
39         while (i < k)
40         {
41             j = i;
42             while (j < k && fabs(a[i].s - a[j].s) < EPS
43                     && fabs(a[i].x - a[j].x) < EPS) j++;
44             r = r < j - i ? j - i : r;
45             i = j;
46         }
47         printf("%d\n", (int)(1.0 + 0.5*sqrt(1.0 + 8.0*r)));
48         scanf("%lld", &n);
49     }
50     return 0;
51 }

```

