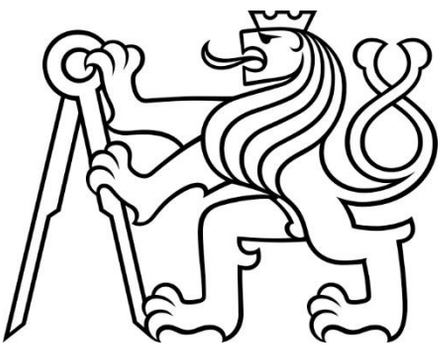
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

GEODEZIE A KARTOGRAFIE



ÚLOHA 2

Algoritmy v digitální kartografii

Konvexní obálky a jejich konstrukce

Katedra geomatiky

Praha 2020

Bc. Eva Frommeltová

Bc. Lucie Hnilicová

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta stavební, Katedra geomatiky
Thákurova 7, 166 29 Praha 6

Obsah

1 Zadání 2
1.2 Údaje o bonusových úlohách 2
2 Popis a rozbor problému 3
3Popis algoritmů 3
3.1 Jarvis Scan 3
3.2 Quick Hull 4
3.3 Sweep Line 5
3.4 Graham Scan 6
4 Problematické situace, generátory 7
4.1 Konstrukce striktní konvexních obálek 7
4.2 Generátor gridu 8
4.3 Generátor kružnice 8
4.4 Generátor náhodných bodů 8
4.5 Generátor elipsy 8
4.6 Generátor čtverce 8
4.7 Generátor obdélníků 9
4.8 Výpočet směrů pro Graham Scan 9
5 Popis Aplikace 9
5.1 Vstupní data 9
5.2 Výstupní data 10
6 Dokumentace 13
6.1 Třída Algorithms 13
6.2 Třída Draw 14
6.3 Třída Generatorforpoint 15
6.4 Pomocné třídy 15
7 Porovnání časové náročnost algoritmů 16
7.1 Jarvis Scan 16
7.2 Quick Hull 18
7.3 Sweep Line 20
7.4 Grafy 23
7.5 Zhodnocení 24
8 Závěr 25

1 Zadání

Vstup: množina $P = \{p_1, ..., p_n\}$, $pi = [x_i, y_i]$.

Výstup: $H(P)$.

Nad množinou P implementujete následující algoritmy pro konstrukci $H(P)$:

- Jarvis Scan,
- Quick Hull,
- Swep Line.

Vstupní množiny bodů včetně vygenerovaných konvexních obálek vhodně vizualizujte. Pro množiny $n \in <1000, 1000000>$ vytvořte grafy ilustrující doby běhu algoritmů pro zvolená n. Měření proveďte pro různé typy vstupních množin (náhodná množina, rastr, body na kružnici) opakovaně (10x) a různá n (nejméně 10 množin) s uvedením rozptylu. Naměřené údaje uspořádejte do přehledných tabulek.

Zamyslete se nad problematikou možných singularit pro různé typy vstupních množin a možnými optimalizacemi. Zhodnoťte dosažené výsledky. Rozhodněte, která z těchto metod je s ohledem na časovou složitost a typ vstupní množiny P nejvhodnější.

1.2 Údaje o bonusových úlohách

Krok	**Řešeno/neřešeno**
Konstrukce konvexní obálky metodou Graham Scan	Ano
Konstrukce striktní konvexních obálek pro všechny uvedené algoritmy.	Ano
Ošetření singulárního případu u Garvis Scan: existence kolineárních bod· v datasetu.	Ano
Konstrukce Minimum Area Enclosing box některou z metod (hlavní směry budov).	Ne
Algoritmus pro automatické generování konvexních/nekonvexních množin bodů různých tvarů (kruh, elipsa, čtverec, star-shaped, popř. další).	Ano

2 Popis a rozbor problému

Cílem úlohy je vytvoření aplikace, která pro *n* vygenerovaných bodů vytvoří konvexní obálku použitím různých algoritmů. Množina bodů byla generována v různých tvarech (grid, kruh, náhodné body). Pro dané algoritmy byla měřena i doba, za kterou konvexní obálku vytvoří. Časové náročnosti jednotlivých algoritmů byly na závěr porovnány.

3 Popis algoritmů

V této kapitole budou popsány jednotlivé algoritmy sloužící pro výpočet konvexní obálky.

3.1 Jarvis Scan

Algoritmus připomíná postup balení dárku do papíru. Lze ho rozšířit i do R3 a má jednoduchou implementaci. Nutný předpoklad je, že v množině *S* nejsou tři kolineární body. Nehodí se pro velké vstupní množiny *S*.

Prvním krokem je nalezení bodu s y_{min}. Následně se postupně vyhledávají body s maximálním úhlem od posledních dvou bodů konvexní obálky. Bod s maximálním úhlem se přidá do obálky. Algoritmus končí ve chvíli, kdy se první a poslední bod rovnají.

Výpočet úhlu ω:

$$u = |p_{j+1}, p_j|$$

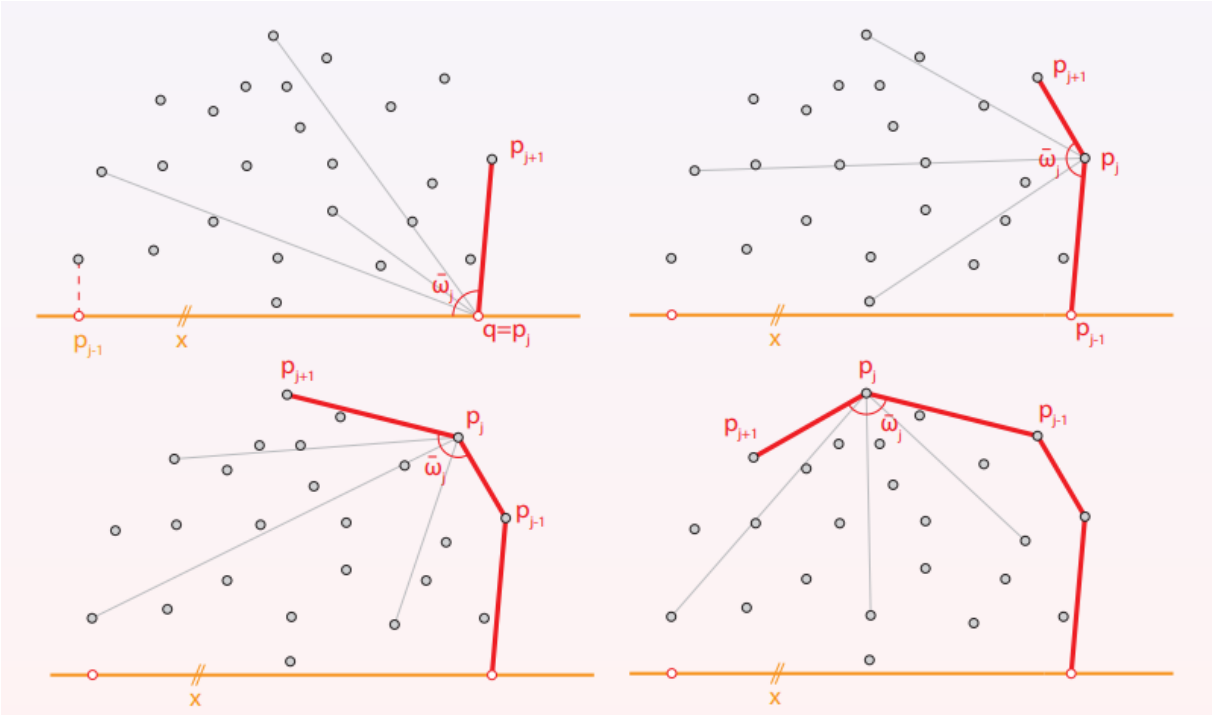
$$v = |p_j, p_i|$$

$$\omega = \left|\text{acos}\frac{u \times v}{|u||v|}\right|,$$

kde p_{j+1}, p_j jsou poslední dva body konvexní obálky *H*, p_i je i-tý bod z množiny bodů.

Jednotlivé kroky algoritmu:

1. Nalezení pivotu *q*, *q = min(y_i)*
2. Přidej *q* → *H*
3. Inicializuj: $p_{j-1} \in X$, $p_j = q$, $p_{j+1} = p_{j-1}$
4. Opakuj, dokud $p_{j+1} \neq q$:
5. Nalezni $p_{j+1} = arg\ max < (p_{j-1}, p_j, p_i)$
6. Přidej p_{j+1} → *H*
7. $p_{j-1} = p_j$; p_j = pj+1



Obrázek 1: Jarvis Scan, zdroj: https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk4.pdf

3.2 Quick Hull

Algoritmus Quich Hull využívá strategii „rozděl a panuj“. Znamená to, že rozdělí danou množinu na horní a dolní část. Vytvoří obálku, kterou postupně konvertuje na konvexní. Ve většině případů je tento algoritmus rychlý. Využívá rekurzi.

V prvním kroku jsou setříděny body podle souřadnice x. Vezme se bod q_1 (x_{min}) a $q_3 = (x_{max})$, zařadí je do konvexní obálky. Mezi těmito dvěma body se vytvoří přímka, která rozdělí množinu na spodní a horní část. Tyto dvě části se řeší odděleně, a nakonec jsou spojeny. Algoritmus obsahuje lokální a globální proceduru. Lokální procedura hledá nejvzdálenější bod od dané přímky, který přidá do konvexní obálky *H*. Dále vzniknout dvě nové přímky, od kterých je napravo od spojnice opět hledán nejvzdálenější bod. Tímto způsobem je postupováno, dokud se napravo od spojnice nachází body.

Vzdálenost bodu od přímky:

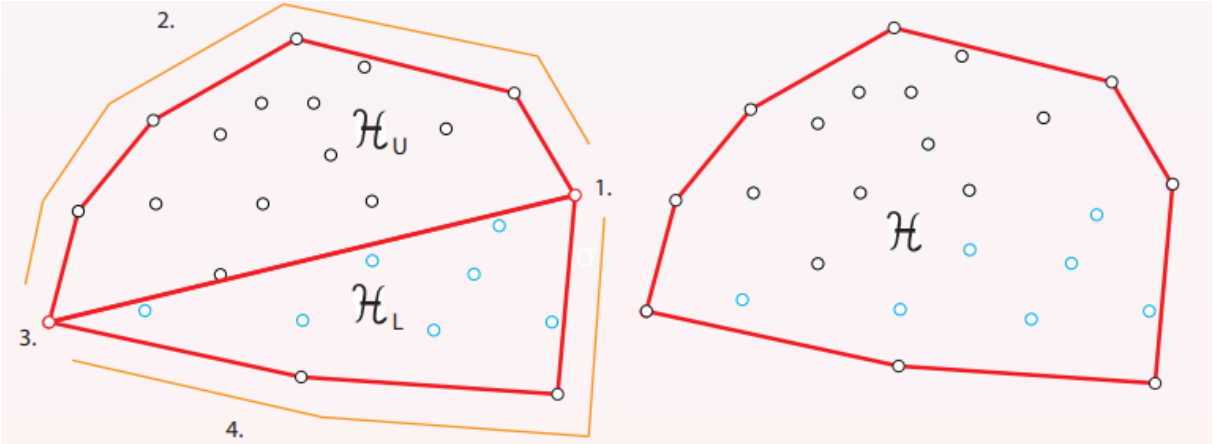
$$d(A,p) = \frac{|x_A(y_1-y_2)+x_1(y_2-y_A)+x_2(y_A-y_1)|}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}},$$

A je bod o souřadnicích $x_A\ a\ y_A$, p je přímka daná body p_1, p_2.

Globální procedura:

1. Nalezení bodů: $q_1 = argmin(x_i), q_3 = argmax(x_i)$
2. Rozdělení na množiny: $S = S_u \cup S_l$
3. $H \leftarrow q_3$
4. H_u - spuštění lokální procedury
5. $H \leftarrow q_1$
6. H_l - spuštění lokální procedury,

kde H je polygon obsahující konvexní obálku, S je množina bodů, S_u je horní část množiny bodů a S_L je dolní část.

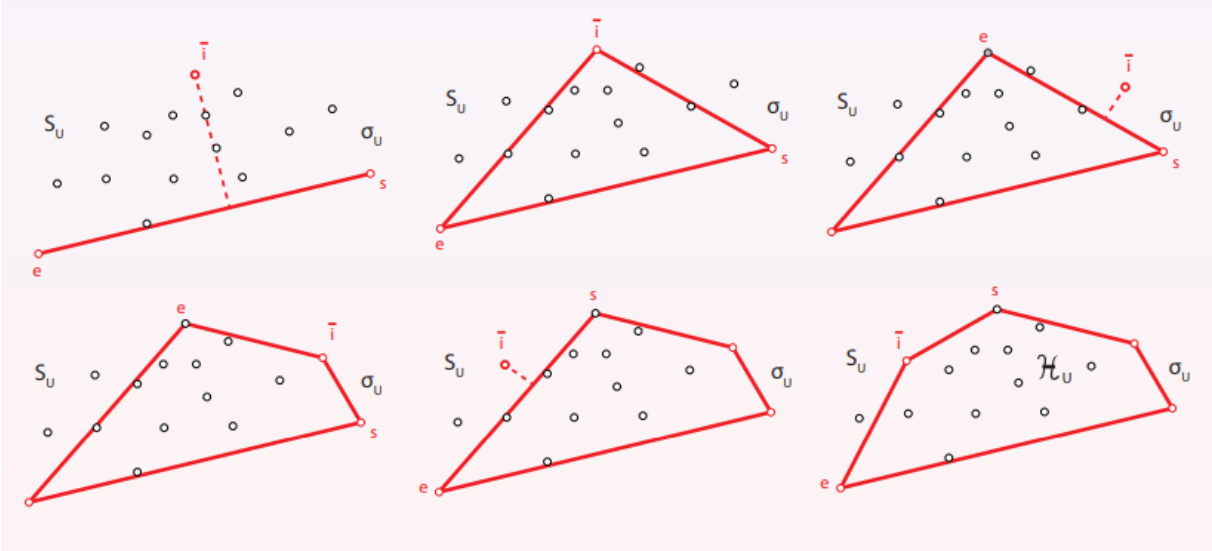


Obrázek 2: Quick Hull, zdroj: https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk4.pdf

Lokální procedura:

1. Hledání bodu $\overline{p} = \arg\max\|pi - (ps, pe)\|, p \in \sigma r\,(ps, pe) - \overline{p}$ nejvzdálenější bod napravo od spojnice ps, pe
2. Pokud $\overline{p} \neq 0$:
   i. Rekurze nad segmentem $(ps, \overline{p})$
   ii. $H \leftarrow \overline{p}$
   iii. Rekurze nad segmentem $(\overline{p}, pe)$,

kde ps je počáteční bod spojnice, pe je koncový bod spojnice.



Obrázek 3: Quick Hull, zdroj: https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk4.pdf

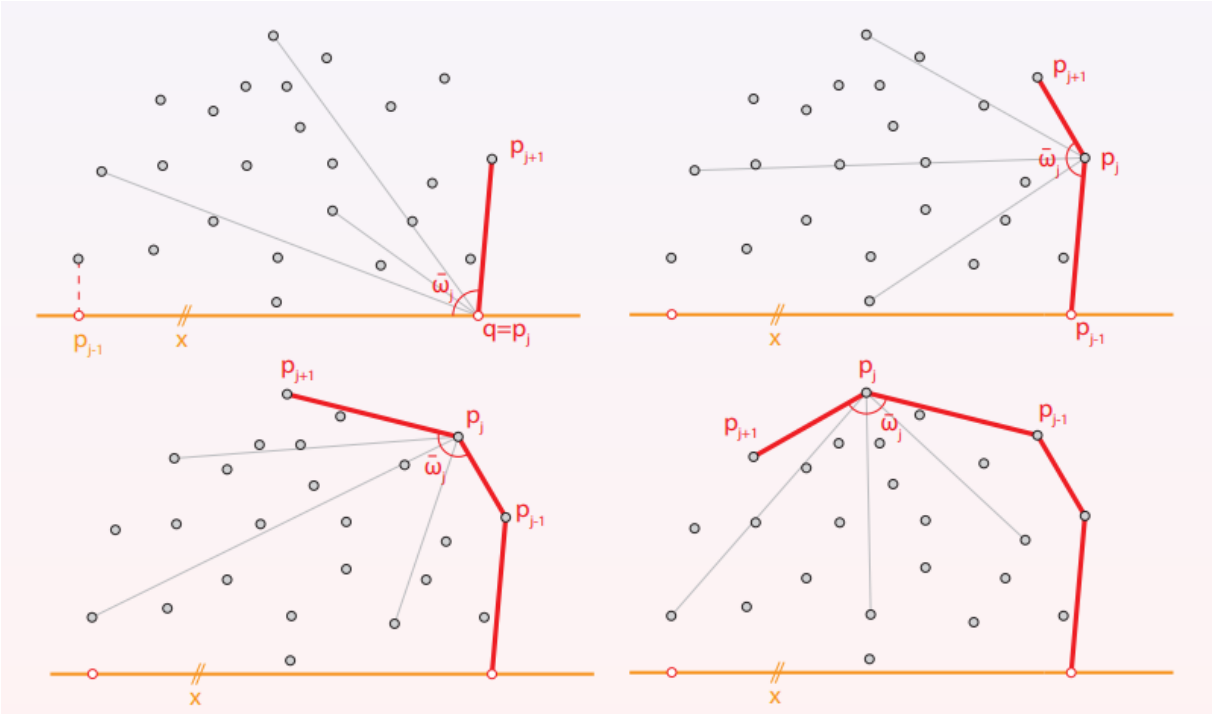
3.3 Sweep Line

Metoda Sweep Line je označována jako metoda zametací přímky, využívá strategii inkrementální konstrukce.

Nejprve se body množiny *S* setřídí podle souřadnice x, dále se množina rozdělí osou x na dvě části – zpracovanou a nezpracovanou část. Začátek algoritmu je řešen pomocí iniciálního dvojúhelníku (trojúhelníku). Algoritmus se dále dělí na dvě iterativní fáze. V první fázi dochází k připojování nových bodů do již vytvořené obálky, tím může dojít k porušení konvexity. V druhé fázi jsou vyloučeny nekonvexní vrcholy z množiny konvexní obálky.

Jednotlivé kroky algoritmu:

1. Setříděni bodů podle x
2. Inicializace vektorů bodů předchůdců p(m) a následníků n(m), m je počet bodů v množině S
3. *Nastavení hodnot: n [0] = 1; n[1] = 0; p[0] = 1; p[1] = 0;*
4. *For cyklus pro všechny body od i=2*
    a) *Když yi>yi-1*
    b) *p[i] = i-1; n[i] = n[i-1]*
    c) *jinak*
    d) *d)p[i] = p[i-1]; n[i] = i–1*
    e) *spojení předchůdce a následníka i: p[n[i]] = i; n[p[i]] = i;*
    f) *Pokud n[n[i]]∈ σ_R(i, n[i]): p[n[n[i]]]=i, n[i]=n[n[i]]*
    g) *Pokud p[p[i]]∈ σ_L(i, p[i]): n[p[p[i]]]=i, p[i]=p[p[i]]*



Obrázek 4: Sweep Line, zdroj: https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk4.pdf

3.4 Graham Scan

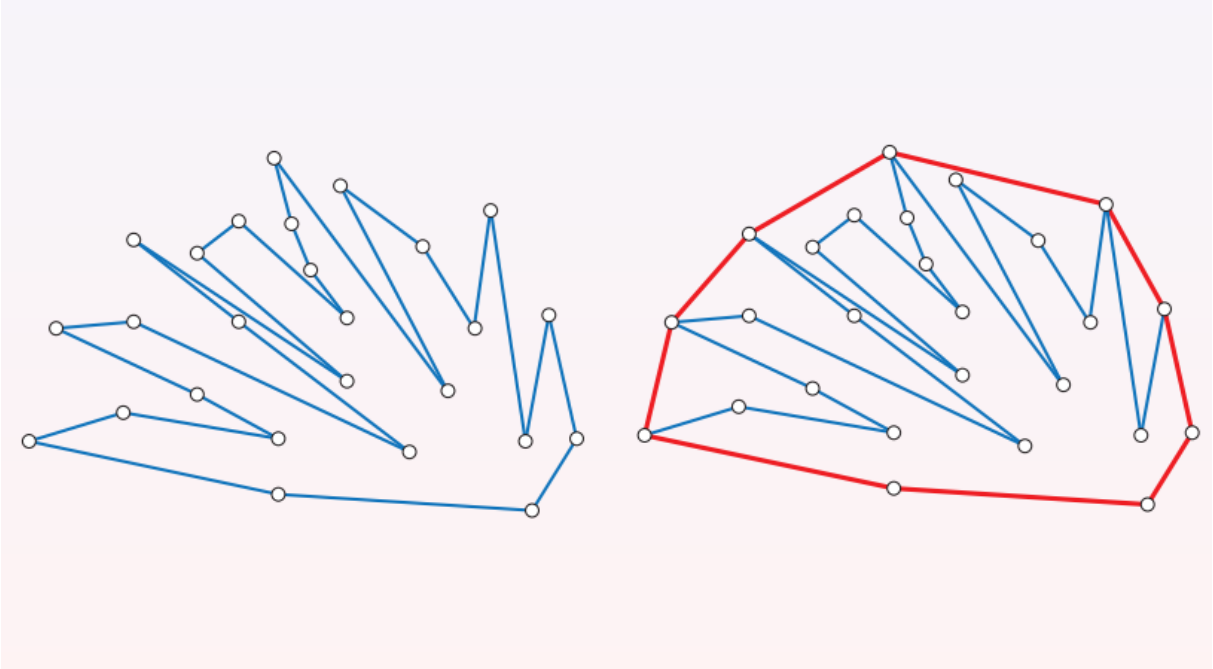
Algoritmus Graham Scan převádí star-shaped polygon na konvexní obálku *H*. Lze ho použít i na rozsáhlé množiny bodů.

Prvním krokem algoritmu je setřídění bodů podle úhlu, který svírají přímka tvořená pivotem *q* a bodem p_i s rovnoběžkou osy x procházející *q*. Jako bod *q* je volen bod, který má minimální souřadnici y. Konvexní obálka obsahuje *q* a bod s maximální úhlem. Dále se vloží do obálky předposlední bod

setříděné množiny a následně je zkoumán další bod, jestli leží nalevo od spojnice posledních dvou bodů přidaných do konvexní obálky. Pokud ano, je přidán do *H*. Pokud ne, tak se bod vyloučí a je postupováno dále.

Jednotlivé kroky algoritmu:

1. Nalezení pilota $q = min(y_i)$
2. Setřídění dle úhlu $\omega = (x, qp_i)$
3. Pokud $\omega_k = \omega$, vymaž bod p_k, p_i bližší ke q
4. Inicializuj j=2, S=∅
5. q a $p_1 \rightarrow H$
6. Opakuj pro j <n:
    a. Když p_i je vlevo od p_{t-1},p_t
    b. Vlož p_j do *H*
    c. j = j+1
    d. jinak vyřaď poslední bod z H



Obrázek 5: Graham Scan, zdroj: https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk4.pdf

4 Problematické situace, generátory

4.1 Konstrukce striktní konvexních obálek

Striktně konvexní obálky jsou takové, že tři po sobě jdoucí body neleží na stejné přímce. Pro tvorbu striktně konvexní obálky byla použita funkce pro určení pozice bodu a přímky. Byly vzaty dva body, které tvoří přímku, následně byl testován následující bod, jestli leží na dané přímce. Pokud leží, je prostřední body přímky z konvexní obálky vyřazen. Body, které náleží striktně konvexní obálce byly zvýrazněny.

4.2 Generátor gridu

Funkce vygeneruje body ve čtvercové síti o celočíselných souřadnicích. Počáteční bod je volen v levém horním rohu hodnotou p (10, 50). Počet bodů v řádcích a sloupcích je reprezentován odmocninou ze zadaného počtu bodů. Rozestup mezi body je volen poměrem výšky okna a počtu bodů.

4.3 Generátor kružnice

Pro generování kružnice byl nejprve vytvořen střed kružnice *s* (x_S, y_S), který byl umístěn ve středu kreslícího plátna. Dále byl určen úhel $\varphi = \frac{2\pi}{n}$ podle, kterého se jednotlivé body natáčejí. Body kružnice jsou postupně vytvořeny s konstantním rozestupem a vzdáleností od poloměru, kde úhel tvořený spojnicemi dvou sousedních bodů se středem kružnice je roven φ. Rovnice kružnice:

$$x = x_S + \left(\frac{h}{2} - 100\right)\cos(\varphi)$$
$$y = y_S + \left(\frac{h}{2} - 100\right)\sin(\varphi),$$

kde h je výška kreslícího plátna.

4.4 Generátor náhodných bodů

Funkce vygeneruje ze zadaného počtu bodů náhodné rozmístění bodů. Náhodné body byly vygenerovány pomocí funkce *rand ()*. Generované souřadnice jsou děleny výškou a šířkou kreslícího plátna, aby nedocházelo k vykreslování i mimo něj.

$$x = rand()modulo\ h$$
$$y = rand()modulo\ w,$$

kde h je výška kreslícího plátna(canvasu) a w je šířka.

4.5 Generátor elipsy

Tato funkce funguje na principu výše zmíněného generování kružnice. Liší se pouze v určení samotných souřadnic, kdy velikosti poloos a, b jsou dány poměrem výšky a šířky okna s počtem bodů. Rovnice elipsy:

$$x = x_S + \left(\frac{w}{2} - 100\right)\cos(\varphi)$$
$$y = y_S + \left(\frac{h}{2} - 100\right)\sin(\varphi),$$

kde h je výška kreslícího plátna(canvasu) a w je šířka.

4.6 Generátor čtverce

Pro generování čtverce byla modifikována funkce generující grid. Počet řádků a sloupců pro generování je převzat ze zadané hodnoty. Do výpočtu byla přidána podmínka, která omezuje vykreslení pouze prvních a posledních řádků a sloupců gridu. Rovnice čtverce:

4.7 Generátor obdélníků

Generátor obdélníku je modifikací funkce pro vykreslení čtverce. Pro vykreslení obdélníku je volen počet sloupců zadanou hodnotou. Počet řádků je volen jako polovina z hodnoty zadaných bodů.

4.8 Výpočet směrů pro Graham Scan

Pro výpočet algoritmu Graham Scan je nutné setřídit úhly $\omega = (osa\ x, qp_i)$. Směry byly vypočítány pomocí funkce:

$$\sigma_1 = atan2(Y_{p1} - Y_q, X_{p1} - X_q)$$
$$\sigma_2 = atan2(Y_{p2} - Y_q, X_{p2} - X_q)$$
$$\sigma_1 < \sigma_2,$$

kde q je pivot, pro který platí $q = min(y_i)$ a p_1, p_2 jsou následující body z množiny S. Tím to způsobem jsou setříděny všechny body z množiny. Pokud se směry rovnají, je pořadí určeno podle toho, který bod je vzdálenější od pivotu q. Délky jsou vypočteny pomocí vzorce:

$$d_1 = (X_{p1} - X_q) \times (X_{p1} - X_q) + (Y_{p1} - Y_q) \times (Y_{p1} - Y_q)$$
$$d_2 = (X_{p2} - X_q) \times (X_{p2} - X_q) + (Y_{p2} - Y_q) \times (Y_{p2} - Y_q)$$
$$(\sigma_1 == \sigma_2\ )\ \&\&\ (d_1 > d_2\ )$$

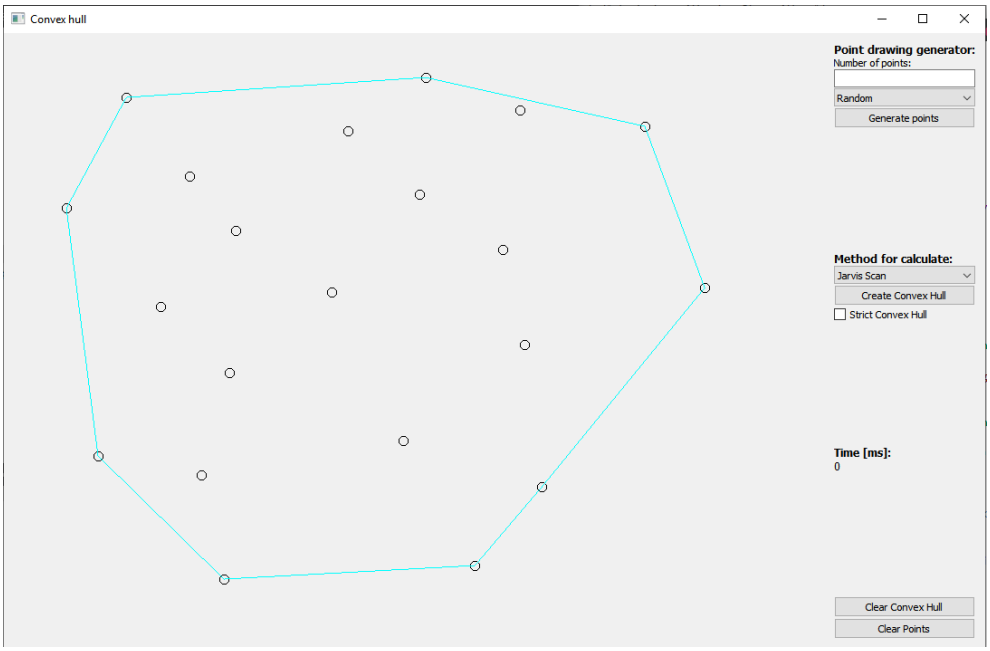
Výpočet směrů je použit i pro odebrání bodů, které mají stejný úhel.

5 Popis Aplikace

V této kapitole je popsána funkcionalita aplikace.

5.1 Vstupní data

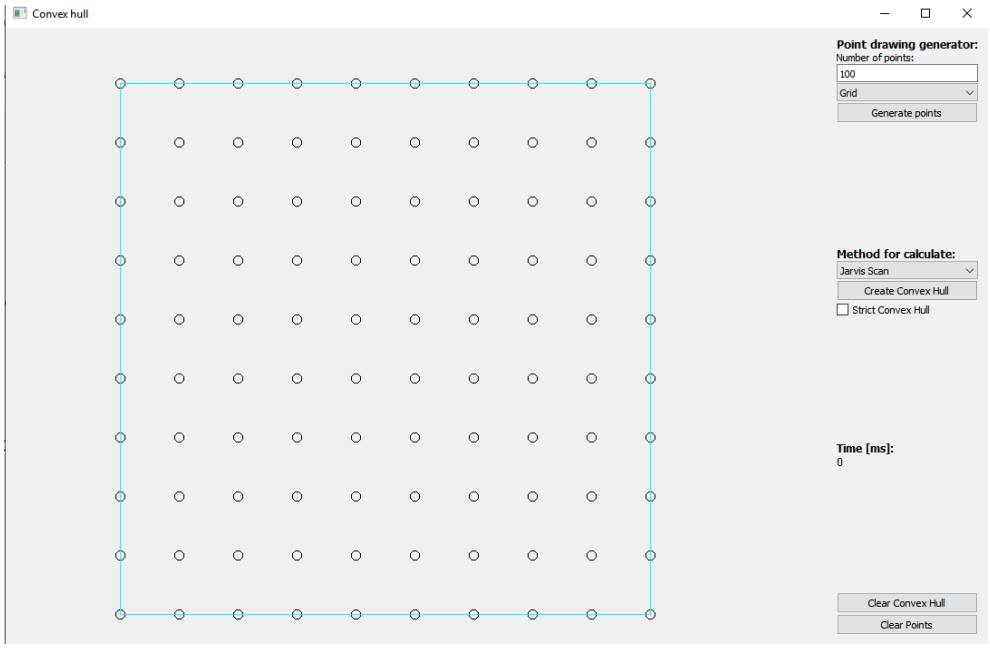
Vstupní data jsou buď vložena uživatelem pomocí myši nebo využitím generátorů bodů. Pokud je využit generátor, je nutné zadat počet bodů *n*. Zadaná hodnota určuje, kolik bodů má množina obsahovat. Dále je uživatelem vybírán typ rozložení bodů z možností – náhodné, body na kružnici, body na elipse, body v gridu, body na čtverci a body na obdélníku. Po kliknutí na tlačítko *Generate points* se body vygenerují.



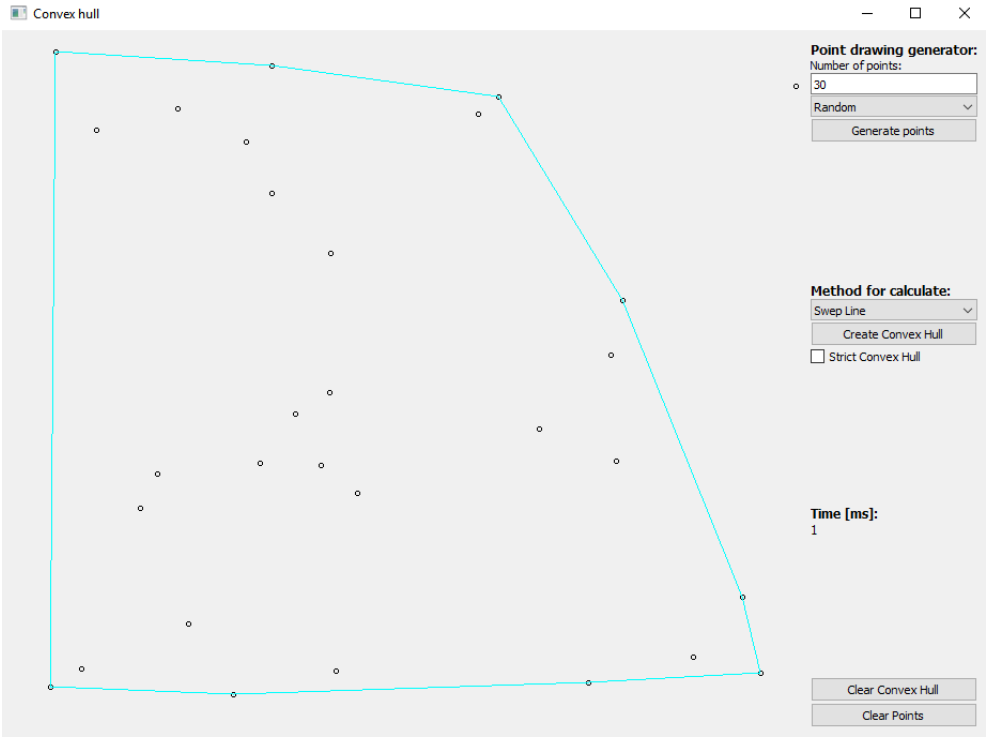
Obrázek 6: Ukázka aplikace – vložení bodů myší a vykreslení konvexní obálky

5.2 Výstupní data

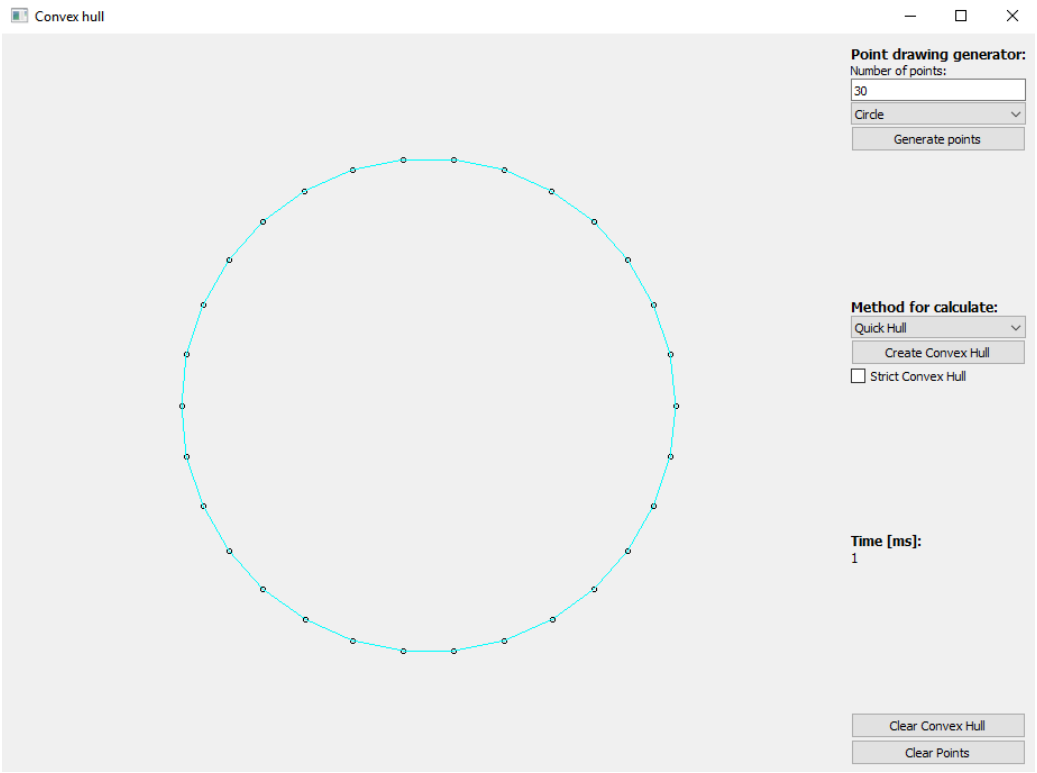
Výstupem programu je grafické znázornění konvexní obálky v kreslícím plátně aplikace. Konvexní obálka je vytvořena algoritmem, který si uživatel zvolí. Po vybrání algoritmu je nutné kliknout na tlačítko *Create Convex Hull*. Díky checkboxu *Strict Convex Hull* je možno vytvořit striktně konvexní obálku. Body, které patří do striktně konvexní obálky, jsou zvýrazněny červeně. Aplikace dále uživateli ukazuje dobu trvaní výpočtu konvexní obálky. Následně si uživatel může pomocí tlačítka *Clear Convex hull* (smazat konvexní obálku) a *Clear Points* (smazat body množiny) vyčistit kreslící plátno.



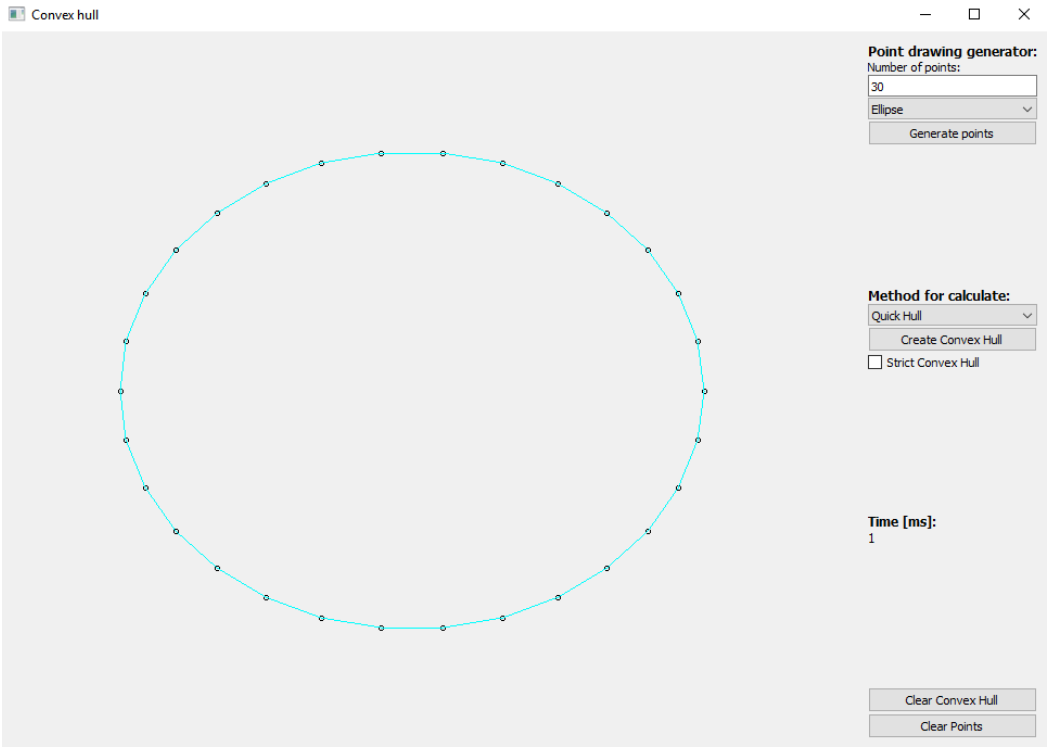
Obrázek 7: Ukázka aplikace – generované body (grid)



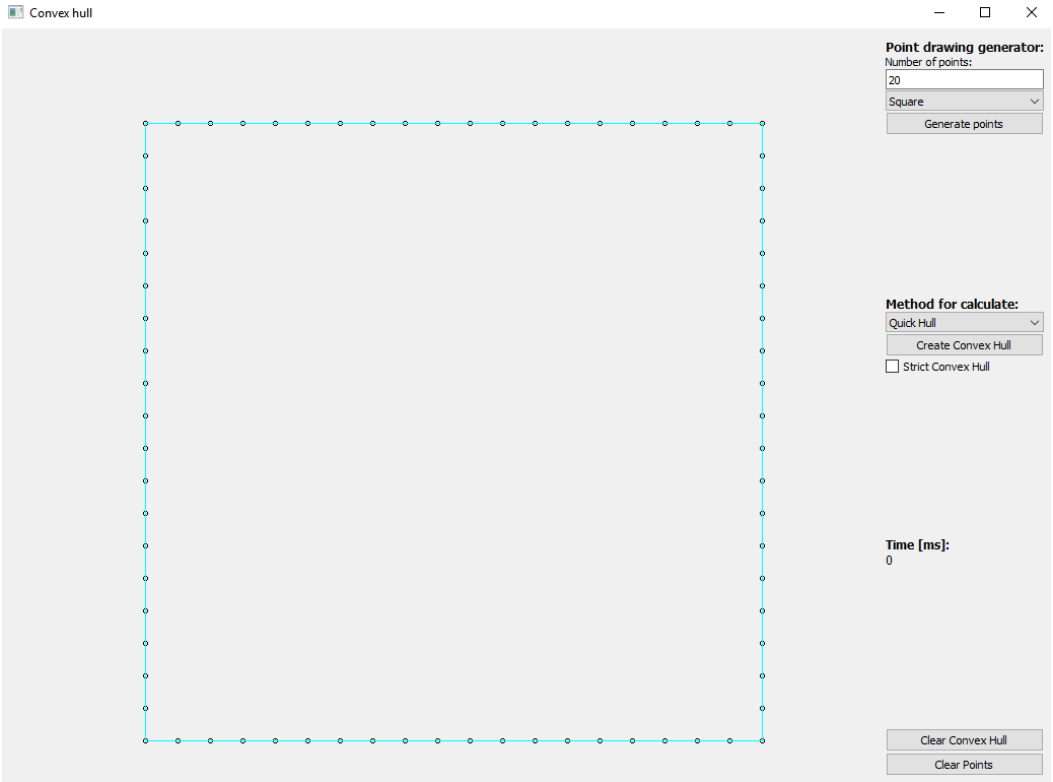
Obrázek 8: Ukázka aplikace – generované body (náhodné)



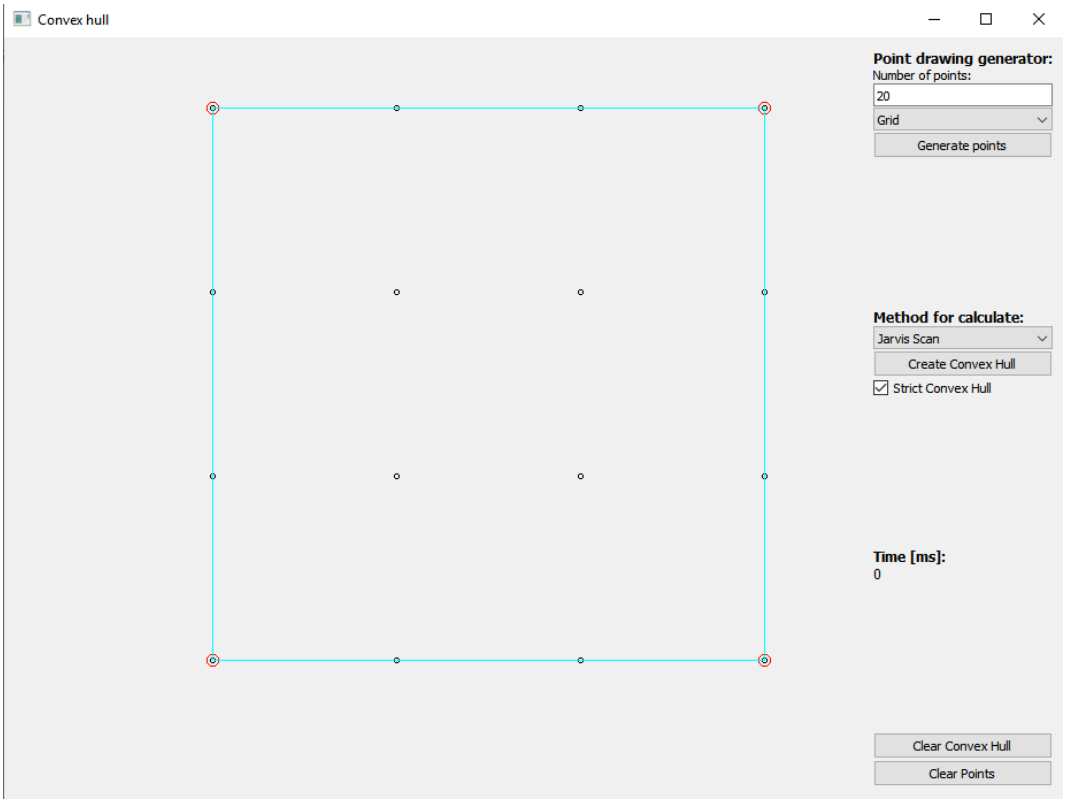
Obrázek 9: Ukázka aplikace – generované body (kružnice)



Obrázek 10:: Ukázka aplikace – generované body (elipsa)



Obrázek 11: Ukázka aplikace – generované body (čtverec)



Obrázek 12: Ukázka aplikace – striktně konvexní obálka

6 Dokumentace

6.1 Třída Algorithms

Třída obsahuje funkce:

```
static double getAngle(QPoint &p1,QPoint &p2,QPoint &p3,QPoint &p4);
```

- Funkce počítá úhel mezi dvěma vektory, vstupní hodnoty jsou body určující vektory

```
static int getPointLinePosition(QPoint &q, QPoint &p1, QPoint &p2);
```

- Funkce určuje polohu bodu vůči přímce, jestli se bod nachází vlevo od přímky (vrací 1), vpravo od přímky (vrací 0), na přímce (vrací -1)

```
static double getPointLineDist(QPoint &a, QPoint &p1, QPoint &p2);
```

- Funkce počítá vzdálenost bodem *a* a přímkou danou body *p1* a *p2*

```
static QPolygon jarvis(std::vector<QPoint> &points);
```

- Funkce sloužící k výpočtu konvexní obálky za pomocí algoritmu Jarvis Scan, vstupem je množina bodů points

```
static QPolygon qhull(std::vector<QPoint> &points);
```

- Funkce sloužící k výpočtu konvexní obálky za pomocí algoritmu Quick Hull, vstupem je množina bodů points

```
static void qh(int s, int e, QPolygon &points, QPolygon &ch);
```

- Funkce pro rekursivní proceduru pro algoritmus Quick Hull, vstupem je množina bodů points

```
static QPolygon sweepLine(std::vector<QPoint> &points);
```
- Funkce sloužící k výpočtu konvexní obálky za pomocí algoritmu Sweep Line, vstupem je množina bodů points

```
static QPolygon graham(std::vector<QPoint> &points);
```
- Funkce sloužící k výpočtu konvexní obálky za pomocí algoritmu Graham Scan, vstupem je množina bodů points

```
static QPolygon strictCHull(QPolygon &ch);
```
- Funkce pro odstranění kolineárních bodů z konvexní obálky

6.2 Třída Draw

Třída obsahuje funkce:

```
private:
    std::vector<QPoint> points;
```
- deklarace proměnné pro množinu bodů

```
    QPolygon ch;
```
- deklarace proměnné pro konvexní obálku

```
public:
    void mousePressEvent(QMouseEvent *e);
```
- Funkce pro snímání souřadnic bodů z kreslícího plátna po kliknutí myši

```
    void paintEvent(QPaintEvent *e);
```
- Funkce pro vykreslení bodů a konvexní obálky

```
    std::vector<QPoint> & getPoint(){return points;}
```
- Funkce pro získání souřadnic bodů

```
    void setCH(QPolygon & ch_){ch=ch_;};
```
- Funkce pro nastavení konvexní obálky

```
    QPolygon & getCH(){return ch;};
```
- Funkce pro získání konvexní obálky

```
    void setPoints(std::vector<QPoint> &pts) {points = pts;}
```
- Funkce pro nastavení souřadnic bodů

```
    std::vector<QPoint> & getstrictPoint(){return strickpoints;}
```
- Funkce pro získání souřadnic bodů, které tvoří striktně konvexní obálku

```
    void setstrictPoints(std::vector<QPoint> &pts) {strickpoints = pts;}
```
- Funkce pro nastavení souřadnic bodů, které tvoří striktně konvexní obálku

6.3 Třída Generatorforpoint

```
static std::vector<QPoint> generatorRandom(int &n, int &w, int &h);
```

- Funkce pro generování náhodných bodů, vstupem je počet bodů n a výška a šířka kreslícího plátna

```
static std::vector<QPoint> generatorGrid(int &n, int &w, int &h);
```

- Funkce pro generování bodů v gridu, vstupem je počet bodů n a výška a šířka kreslícího plátna

```
static std::vector<QPoint> generatorCircle(int &n, int &w, int &h);
```

- Funkce pro generování bodů na kružnici, vstupem je počet bodů n a výška a šířka kreslícího plátna

```
static std::vector<QPoint> generatorEllipse(int &n, int &w, int &h);
```

- Funkce pro generování bodů na elipse, vstupem je počet bodů n a výška a šířka kreslícího plátna

```
static std::vector<QPoint> generatorSquare(int &n, int &w, int &h);
```

- Funkce pro generování bodů na čtverci, vstupem je počet bodů n a výška a šířka kreslícího plátna

```
static std::vector<QPoint> generatorRectangle(int &n, int &w, int &h);
```

- Funkce pro generování bodů na obdélníku, vstupem je počet bodů n a výška a šířka kreslícího plátna

6.4 Pomocné třídy

sortByAngle

- Třída setřídí úhly podle velikosti, pokud se směry rovnají vrátí bod, který je vzdálenější od pivotu q

sortByX

- Třída setřídí body podle souřadnice x

sortByY

- Třída setřídí body podle souřadnice y

removeByAngle

- Třída setřídí vyhledá směry, které jsou si rovny

sortByYRight

- Třída setřídí body podle nejmenší hodnoty souřadnice y, která se nachází nejvíce vpravo

7 Porovnání časové náročnost algoritmů

V této kapitole se nachází výsledky testování časové náročnosti jednotlivých algoritmů. Algoritmy byly testovány pro počet bodů *n* rovný 1 000, 2 000, 5 000, 10 000, 20 000, 50 000, 100 000, 200 000, 500 000, 1 000 000. Pro každé *n* byl algoritmus spuštěn desetkrát.

7.1 Jarvis Scan

Jarvis Scan													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		1000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	3	2	3	3	4	5	4	4	4	3	3,50	0,72
	Grid	7	6	6	6	6	6	6	7	6	7	6,30	0,23
	Random	2	3	3	2	4	3	3	3	3	2	2,80	0,40

Jarvis Scan													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		2000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	8	8	7	8	8	9	9	8	7	8	8,00	0,63
	Grid	14	17	15	15	18	15	15	16	16	17	15,80	1,17
	Random	6	5	5	5	5	6	6	7	7	8	6,00	1,00

Jarvis Scan													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		5000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	20	19	18	21	18	22	18	19	18	18	19,10	1,37
	Grid	69	68	93	67	87	77	98	68	70	73	77,00	10,90
	Random	13	14	15	18	18	19	17	15	20	12	16,10	2,55

Jarvis Scan													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		10000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	42	38	39	39	38	38	37	38	38	42	38,90	1,64
	Grid	186	182	179	183	181	182	180	180	181	183	181,70	1,90
	Random	37	38	40	44	30	38	37	38	27	40	36,90	4,68

Jarvis Scan													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		20000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	74	75	75	75	75	74	85	75	76	75	75,90	3,08
	Grid	457	469	461	458	461	458	471	463	457	461	461,60	4,63
	Random	103	104	79	115	71	81	75	91	107	88	91,40	14,33

Jarvis Scan													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		50000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	199	186	192	189	186	191	186	184	186	185	188,40	4,32
	Grid	1866	1864	1909	1911	1869	1948	1874	1915	1876	1867	1889,90	27,32
	Random	378	348	334	293	333	381	338	390	337	330	346,20	27,77

Jarvis Scan													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		100000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	447	492	462	440	447	465	451	415	510	459	458,80	25,27
	Grid	6292	6308	6123	6316	6220	6651	6142	6229	6343	6799	6342,30	205,98
	Random	1326	1124	1132	1237	1464	1288	1330	1527	1364	1231	1302,30	123,34

Jarvis Scan													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		200000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	928	929	878	996	926	942	881	898	820	865	906,30	46,21
	Grid	15438	15175	15355	15273	15036	15149	15009	15326	15377	15311	15244,90	138,55
	Random	4726	4921	3602	4104	4149	3686	4292	4067	4317	4238	4210,20	383,46

Jarvis Scan													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		500000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	2038	2351	2028	1915	1995	2060	2011	2064	1975	2324	2076,10	137,18
	Grid	57410	55852	56439	56410	57621	55403	56959	56557	56807	56381	56583,90	630,22
	Random	20275	22125	23648	25890	28198	21739	21125	20991	20350	22425	22676,60	2437,86

Jarvis Scan													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		1000000											
Počet testů testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	4677	4699	4545	4634	4415	4589	4409	5210	4649	4762	4658,90	213,73
	Grid	165504	160538	162156	176947	197154	193515	192309	187720	195382	185188	181641,30	13569,23
	Random	60819	69067	65630	62169	68445	79352	75483	70488	71481	77349	70028,30	5847,69

7.2 Quick Hull

Quick hull													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		1000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	0	0	0	1	0	0	0	0	1	1	0,30	0,46
	Grid	0	1	0	0	0	1	0	1	0	0	0,30	0,46
	Random	0	0	1	0	1	0	0	0	1	0	0,30	0,46

Quick hull													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		2000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	0	0	2	0	1	1	0	0	2	0	0,60	0,80
	Grid	0	1	0	0	2	0	0	1	0	1	0,50	0,67
	Random	0	1	0	0	1	0	1	1	0	0	0,40	0,49

Quick hull													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		5000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	3	6	2	4	1	1	3	1	2	1	2,40	1,56
	Grid	0	0	1	0	0	0	1	0	1	0	0,30	0,46
	Random	2	1	1	1	2	1	2	1	1	1	1,30	0,46

Quick hull													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		10000											
Počet testů testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	2	2	3	2	2	3	2	2	2	2	2,20	0,40
	Grid	1	1	1	1	2	1	1	0	1	1	1,00	0,45
	Random	2	2	2	3	2	2	3	2	2	2	2,20	0,40

Quick hull													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		20000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	6	4	4	4	10	5	4	14	3	7	6,10	3,27
	Grid	2	2	3	2	3	2	2	2	3	2	2,30	0,46
	Random	10	4	6	5	4	11	5	5	4	4	5,80	2,44

Quick hull													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		50000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	10	11	14	20	13	9	20	9	14	24	14,40	4,96
	Grid	4	4	4	5	4	4	4	11	5	7	5,20	2,14
	Random	12	22	15	11	10	33	17	24	9	12	16,50	7,28

Quick hull													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		100000											
Počet testů testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	22	19	19	29	24	21	30	20	27	19	23,00	4,05
	Grid	8	14	15	8	9	10	13	9	17	9	11,20	3,09
	Random	21	31	43	24	22	24	20	28	34	23	27,00	6,83

Quick hull													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		200000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	73	52	49	53	61	52	57	53	56	70	57,60	7,64
	Grid	35	19	18	25	23	24	22	30	18	19	23,30	5,29
	Random	54	44	38	47	39	43	40	37	42	63	44,70	7,72

Quick hull													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		500000											
Počet testů testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	154	94	111	153	129	179	112	119	105	106	126,20	25,82
	Grid	60	66	76	54	58	44	61	56	70	57	60,20	8,45
	Random	82	74	85	89	116	110	94	115	89	88	94,20	13,75

Quick hull													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		1000000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	261	233	306	205	272	235	261	207	231	220	243,10	30,06
	Grid	89	90	88	84	117	105	128	192	101	102	109,60	30,44
	Random	168	202	285	162	253	170	151	186	174	176	192,70	40,89

7.3 Sweep Line

Sweep line													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		1000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	0	0	0	0	1	0	0	0	0	0	0,10	0,30
	Grid	0	0	0	0	0	0	1	0	0	0	0,10	0,30
	Random	0	1	0	0	0	0	1	0	0	0	0,20	0,40

Sweep line													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		2000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	0	0	1	0	0	1	0	0	0	0	0,20	0,40
	Grid	0	1	0	0	0	0	0	0	0	0	0,10	0,30
	Random	0	0	0	0	1	0	0	0	0	1	0,20	0,40

Sweep line													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		5000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	0	0	1	0	0	0	0	1	1	0	0,30	0,46
	Grid	0	0	0	0	1	1	1	0	0	1	0,40	0,49
	Random	1	0	1	1	1	1	1	0	0	1	0,70	0,46

Sweep line													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		10000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	1	0	0	1	0	1	1	1	1	1	0,70	0,46
	Grid	2	2	1	2	1	1	2	2	1	1	1,50	0,50
	Random	2	1	2	2	1	1	2	1	1	1	1,40	0,49

Sweep line													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		20000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	1	1	2	1	1	2	1	1	1	1	1,20	0,40
	Grid	3	3	3	3	2	2	3	2	3	3	2,70	0,46
	Random	3	3	3	3	3	3	3	3	3	3	3,00	0,00

Sweep line													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		50000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	2	2	3	2	3	2	2	2	2	3	2,30	0,46
	Grid	7	7	7	7	6	6	7	7	7	7	6,80	0,40
	Random	8	8	8	7	8	8	8	7	8	8	7,80	0,40

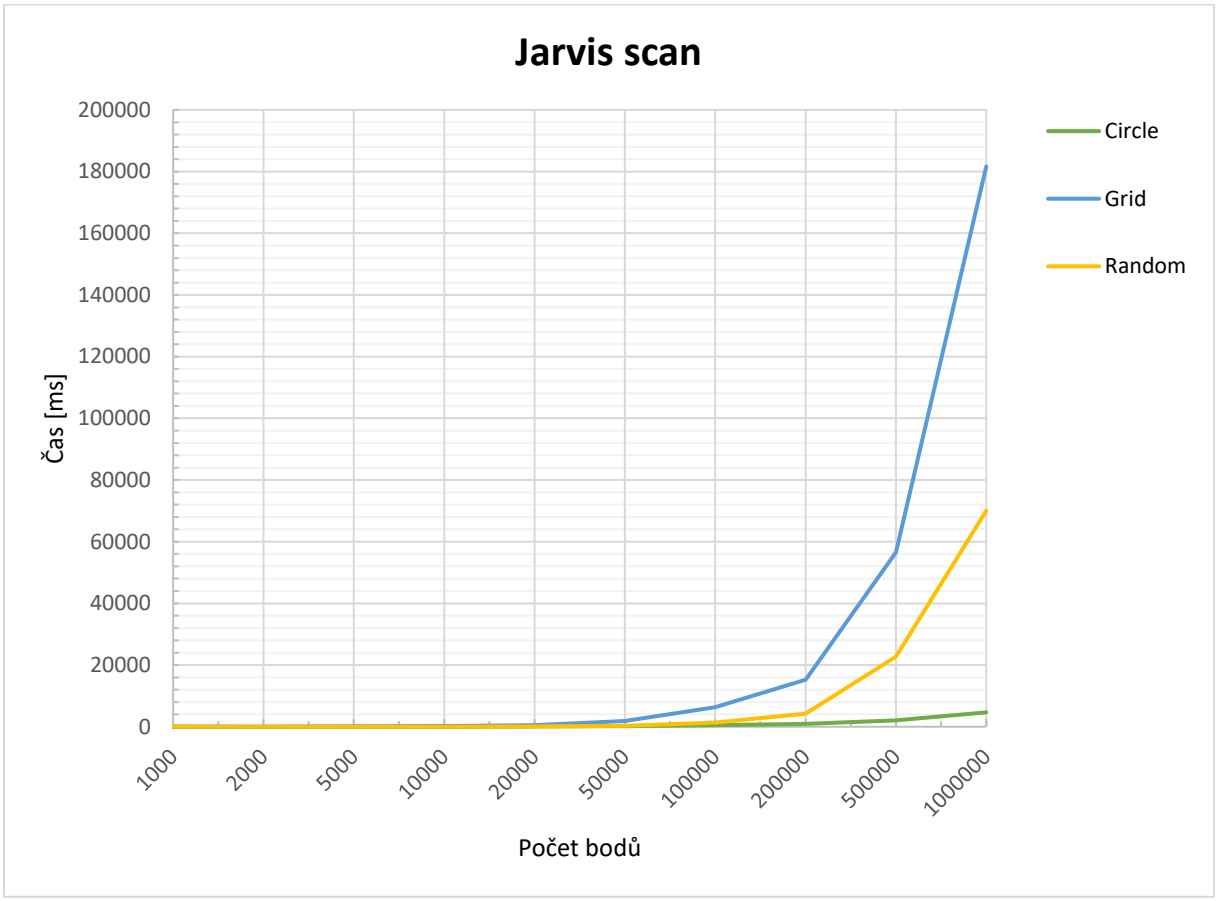
Sweep line													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		100000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	5	4	4	5	4	4	4	5	6	5	4,60	0,66
	Grid	14	14	15	14	15	13	14	15	14	14	14,20	0,60
	Random	16	15	16	16	15	16	16	18	16	16	16,00	0,77

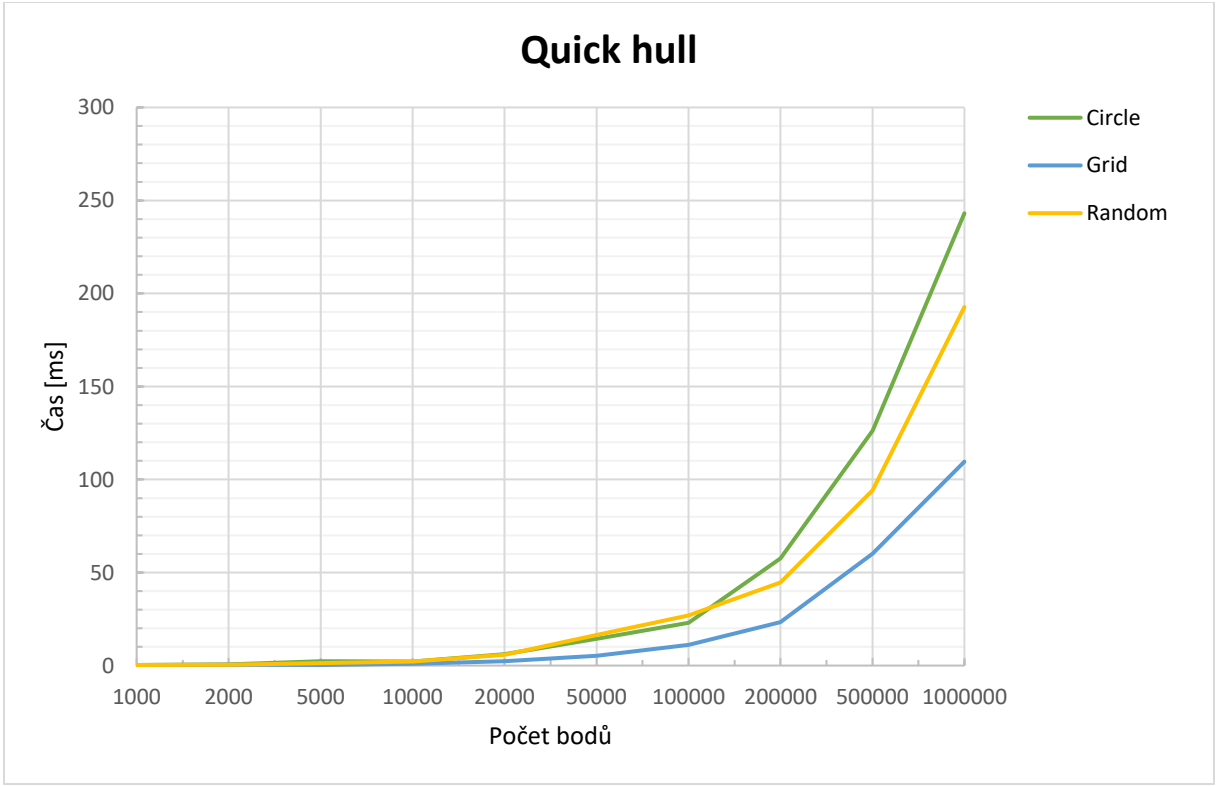
Sweep line													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		200000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	9	8	9	8	8	8	9	9	9	8	8,50	0,50
	Grid	28	29	28	28	28	29	30	29	28	29	28,60	0,66
	Random	32	32	32	33	32	32	31	32	31	32	31,90	0,54

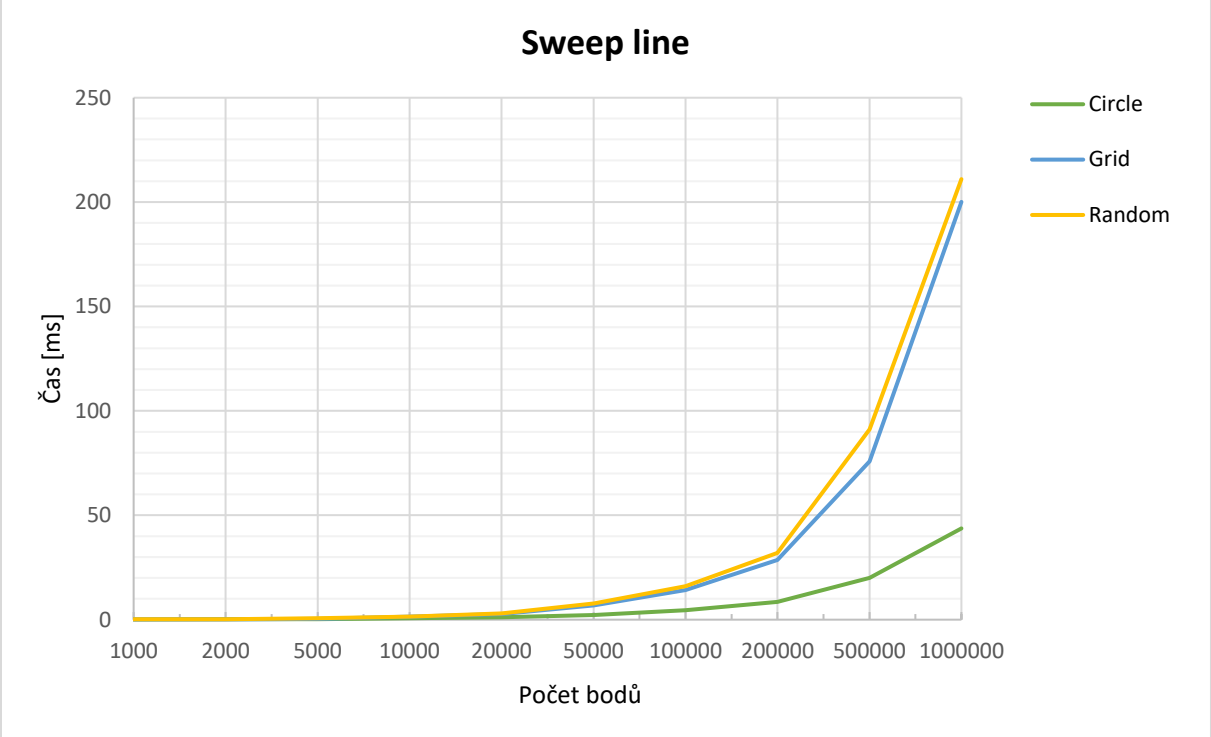
Sweep line													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		500000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	21	20	20	20	19	20	20	20	20	20	20,00	0,45
	Grid	89	73	72	72	76	79	74	79	73	72	75,90	5,07
	Random	81	83	80	88	100	93	94	103	88	102	91,20	8,13

Sweep line													
		t [ms]										Průměr [ms]	Rozptyl[ms]
Počet bodů [n]		1000000											
Počet testů		1	2	3	4	5	6	7	8	9	10		
Typ testu	Circle	47	48	42	43	42	42	44	42	45	42	43,70	2,15
	Grid	186	211	186	199	153	198	148	227	232	261	200,10	33,02
	Random	248	259	174	217	212	203	241	199	175	182	211,00	28,85

7.4 Grafy


Jarvis scan
200000
180000
160000
140000
120000
100000
80000
60000
40000
20000
0
Čas [ms]
1000
2000
5000
10000
20000
50000
100000
200000
500000
1000000
Počet bodů
Circle
Grid
Random


Quick hull
300
250
200
150
100
50
0
Čas [ms]
1000
2000
5000
10000
20000
50000
100000
200000
500000
1000000
Počet bodů
Circle
Grid
Random



7.5 Zhodnocení

Časová náročnost algoritmů

Časová náročnost byla testována u třech – Jarvis Scan, Quick Hull, Sweep Line.

Jarvise scan nejdéle počítal konvexní obálku bodů na gridu, kde i časová náročnost rychle roste s počtem bodů. Nejlépe si Jarvis poradil s body na kružnici. Pro všechny množiny má největší časovou náročnost.

Pro Quick Hull nejsou tak patrné rozdíly mezi jednotlivými množinami bodů, časová náročnost má podobnou tendenci u všech množin. Jeví se, jako nejvýhodnější algoritmus pro body na gridu a pro náhodné body.

Sweep Line si velice dobře poradí s body na kružnici, kde časová náročnost příliš nestoupá. Body na gridu a náhodné body mají velice podobnou křivku. Dosáhl nejlepších časů pro body na kružnici.

8 Závěr

Dle zadání byla vytvořena aplikace pro tvorbu konvexní obálky nad množinou bodů. Pro tvorbu obálky byly použity čtyři různé algoritmy – Jarvis Scan, Quick Hull, Sweep Line a Graham Scan.

Časová náročnost byla testována u třech – Jarvis Scan, Quick Hull, Sweep Line. Jako nejvýhodnější se pro body na gridu a pro náhodné body jeví Quick Hull. Pro body na kružnici je nejvýhodnější využit Sweep Line.

Pro čtvrtí algoritmus Graham Scan nebyla testována časová náročnost vykreslení konvexní obálky.

Seznam obrázku

Obrázek 1: Jarvis Scan, zdroj: https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk4.pdf 4
Obrázek 2: Quick Hull, zdroj: https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk4.pdf 5
Obrázek 3: Quick Hull, zdroj: https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk4.pdf 5
Obrázek 4: Sweep Line, zdroj: https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk4.pdf 6
Obrázek 5: Graham Scan, zdroj: https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk4.pdf 7
Obrázek 6: Ukázka aplikace – vložení bodů myší a vykreslení konvexní obálky 10
Obrázek 7: Ukázka aplikace – generované body (grid) 10
Obrázek 8: Ukázka aplikace – generované body (náhodné) 11
Obrázek 9: Ukázka aplikace – generované body (kružnice) 11
Obrázek 10:: Ukázka aplikace – generované body (elipsa) 12
Obrázek 11: Ukázka aplikace – generované body (čtverec) 12
Obrázek 12: Ukázka aplikace – striktně konvexní obálka 13