ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA STAVEBNÍ, OBOR GEODÉZIE A KARTOGRAFIE
KATEDRA GEOMATIKY

Název předmětu

ALGORITMY DIGITÁLNÍ KARTOGRAFIE A GIS

Úloha	Název úlohy				
2	Konvexní obálky a jejich konstrukce				
Akademický rok	**Semestr**	**Studijní skupina**	**Vypracoval**	**Datum**	**Klasifikace**
2018/2019	3.	60	Janovský Michal karving47@gmail.com	07. 11. 2018	

1 Obsah

2 Seznam obrázků

1 Zadání úlohy

Úloha č. 2: Geometrické vyhledávání bodu

Vstup: $P - \{p_1, \ldots\ldots, p_n\}, p_i - [x, y_i]$
Výstup: H(P)

Nad množinou P implementujte následující algoritmy pro konstrukci H(P):

- Jarvis Scan
- Quick Hull
- Swep Line

Vstupní množiny bodů včetně vygenerovaných konvexních obálek vhodně vizualizujte. Pro množiny $n \in \langle 1\,000, 1\,000\,000 \rangle$ vytvořte grafy ilustrující doby běhu algoritmů pro zvolená n. Měření proveďte pro různé typy vstupních množin (náhodná, rastr, kružnice) 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 uvedených metod je s ohledem na časovou složitost a typ vstupní množiny P nejvhodnější.

Povinná část úlohy:

- Konstrukce konvexních obálek metodami Jarvis Scan, Quick Hull, Sweep Line

Bonusové úlohy:

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

2 Údaje o bonusových úlohách

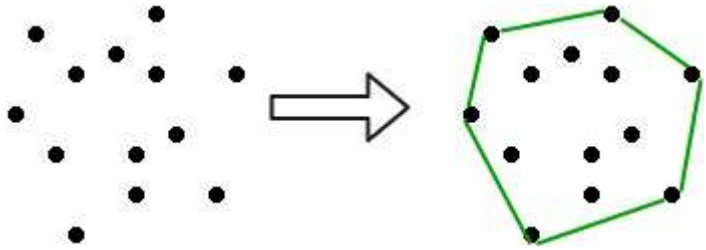
V rámci úlohy byly implementovány bonusové úlohy:

- Konstrukce konvexní obálky metodou Graham Scan
- Konstrukce striktně konvexních obálek pro všechny uvedené algoritmy (i Graham Scan)
- Ošetření singulárního případu u Jarvis Scan: existence kolineárních bodů v datasetu
- Algoritmus pro automatické generování konvexních/nekonvexních množin bodů různých tvarů

3 Popis a rozbor problému + vzorce

Konvexní obálka množiny n bodů S je nejmenší množina bodů, která množinu S obsahuje.

Dáno: Množina n bodů $S = \{p_1, \dots, p_n\}\ v\ R^2$, kde $p_i = [x_i, y_i]$
Hledáme: Nejmenší konvexní množinu $C, \forall p_i \in C$



Obrázek 1 - Konvexní obálka

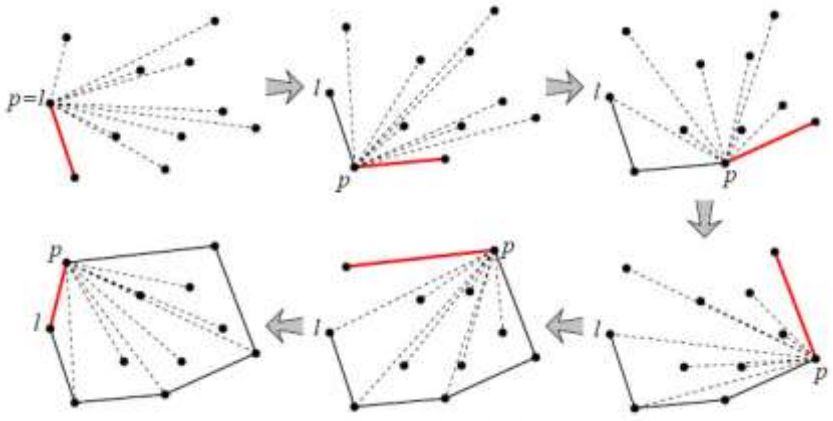
Úkolem této úlohy je konvexní obálku množiny n bodů vytvořit, a to algoritmy Jarvis Scan, Quick Hull, Sweep Line a pro jednotlivé algoritmy změřit jejich časovou náročnost a porovnat je.

4 Popisy algoritmů formálním jazykem

Metody použité v této úloze, byly implementovány v programovacím jazyce C++ v prostředí Qt Creator. V TZ jsou popsané pouze použité metody, nicméně se nevylučuje existence mnoha dalších metod.

4.1 Jarvis Scan

Algoritmus Jarvis Scan funguje na principu hledání maximálních úhlů. Jako první se nalezne tzv. pivot, což je počáteční bod, který zaručeně leží v konvexní obálce množiny (v našem případě Y = min). poté se hledají 2 další body, kde úhel svíraný mezi pivotem a těmito body je maximální. Po nalezení těchto bodů se posune hledání na další bod a postup se opakuje. Problémy, které musí být u algoritmu Jarvis Scan ošetřeny jsou: Nalezení více pivotů a body v jedné přímce.



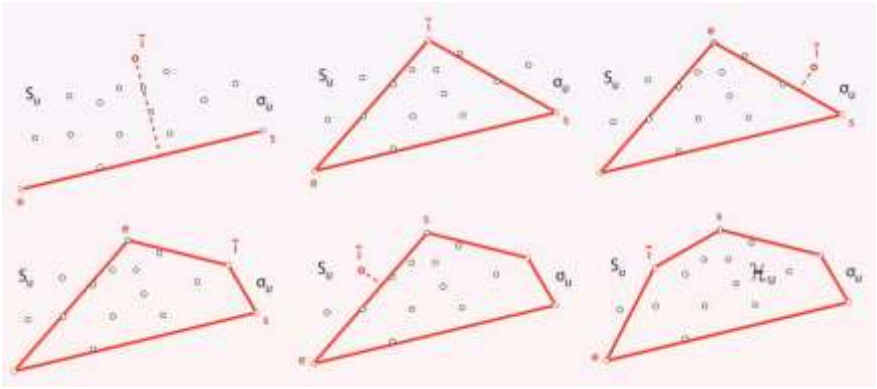
Obrázek 2 - Znázornění Jarvis Scan

Postup výpočtu:

1) Nalezení pivota $q = \min(y_i), \max(x_i)$
2) Vložení pivota do obálky H
3) Inicializace $p_j = q;\ p_{j+1} = p_{j-1}$
4) Dokud $p_{j+1} \neq q$
   - $p_{j+1} = argmax_{\forall p_i \in p} \angle(p_{j-1}; p_j; p_i)$
   - $p_{j+1} \longrightarrow H$
   - $p_{j-1} = p_j;\ p_j = p_{j+1}$

4.2 Quick Hull

Algoritmus Quick Hull využívá techniky „rozděl a panuj“, kdy množinu bodů rozdělí na 2 části, přičemž každá se zpracovává samostatně. Dochází k výběru 2 bodů s extrémními souřadnicemi X, které slouží jako dělící úsečka. Od této úsečky se hledá nejvzdálenější bod (v obou polovinách množiny), který se zařadí do konvexní obálky. Vznikne nová úsečka a hledání nejvzdálenějšího bodu se opakuje.



Obrázek 3 - Znázornění Quick Hull

Postup výpočtu:

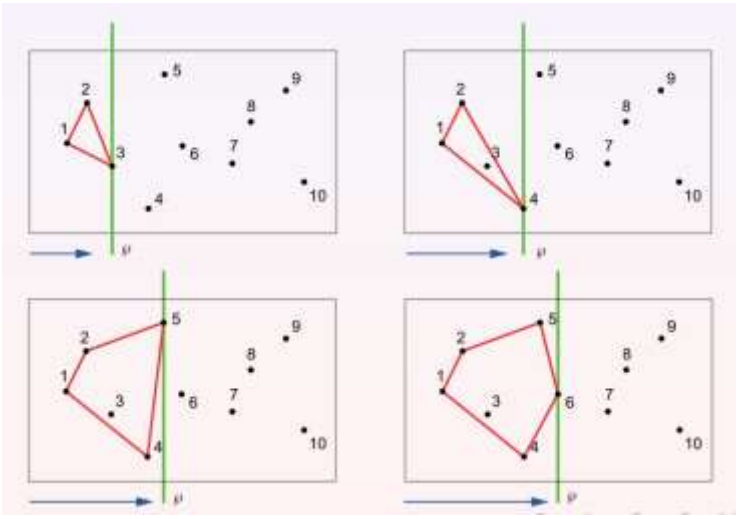
1) Inicializace CH a její horní a dolní části $H = 0;\ S_U = 0;\ S_L = 0$
2) Nalezení extrémních hodnot $q_1 = \min(x_i)\ ; q_2 = \max(x_i)$
3) Přidání $q_1; q_2\ \longrightarrow S_U;\ S_L$
4) Pro všechny $\forall p_i \in S$ rozhodni, zda bod patří do horní části
   - $if\left(p_i \in \sigma_1(q_1;\ q_3)\right) \Longrightarrow p_i \longrightarrow S_U$
   - $else\ p_i \longrightarrow S_L$
5) $q_3 \longrightarrow H$
6) $nalezení\ nejvzdálenějšího\ bodu\ v\ horní$ části $od\ přímky, přidat\ do\ obálky\ a\ opakovat\ vůči\ nové\ straně$
7) $q_1 \longrightarrow H$
8) Opakuj pro dolní část

4.3 Sweep Line

Sweep Line je algoritmus založený na inkrementální konstrukci. Množina bodů je rozdělena na zpracovanou a nezpracovanou část. Souřadnice se seřadí podle jedné ze souřadnic a postupně se bod po bodu vyhodnocují body z nezpracované části. Body se vyhodnocují v závislosti na jejich poloze vůči tečně spojnic zpracovaných bodů. Na začátek se první 3 body označují za vyhodnocené (a tvoří trojúhelník). Algoritmus je citlivý na duplicitní body, proto je třeba množinu bodů vyfiltrovat.

Postup výpočtu:

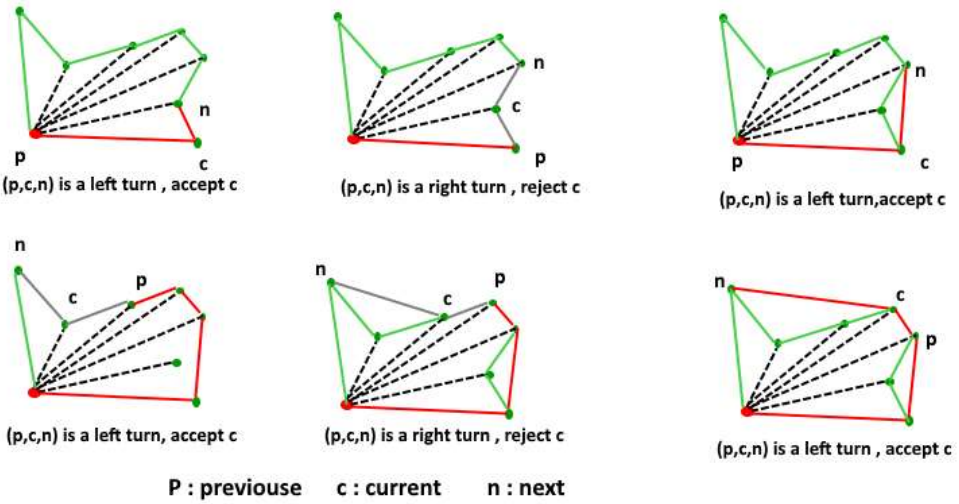
1) $SortP_x = sort(P)\ by\ x$
2) $if\left(p_3 \in \sigma_L(p_1; p_2)\right)$
   $n[1] = 2; n[2] = 3; n[3] = 1$
   $p[1] = 3; p[2] = 1; p[3] = 2$
3) $if\left(p_3 \in \sigma_P(p_1; p_2)\right)$
   $n[1] = 3; n[3] = 2; n[2] = 1$
   $p[1] = 2; p[3] = 1; p[2] = 3$
4) $for\ p_i \in P_S; i > 3$
   $if(y_i > y_{i-1})$
   $p[i] = i - 1; n[i] = n[i - 1]$
   $p[i] = p[i - 1]; n[i] = i - 1$
5) $n[p[i]] = i; p[n[i]] = i$
6) $while(n[n[i]])\ \in \sigma_R(i;\ n[i])$
   $p[n[n[i]]] = i; n[i] = n[n[i]]$
7) $while(p[p[i]])\ \in \sigma_L(i;\ p[i])$
   $n[p[p[i]]] = i; p[i] = p[p[i]]$



Obrázek 4 - Znázornění Sweep Line

4.4 Graham Scan

Algoritmus Graham Scan funguje na principu levotočivosti (CCW), tedy že každá trojice po sobě jdoucích bodů konvexní obálky splňuje kritérium levotočivosti. Nejprve je množina seřazena dle souřadnice Y. Nalezneme pivot a z něj směrníky na všechny ostatní body množiny. Množina je posléze znovu setříděna, tentokrát podle velikosti směrníku. Po setřídění jsou testovány vždy 2 poslední body CH a následující bod ze setříděné množiny. Pokud jsou nalezeny 2 body se stejným směrníkem, pak pokud do CH přidáme ten nejvzdálenější z nich, dostaneme striktně konvexní obálku.



Obrázek 5 - Znázornění Graham Scan

Postup výpočtu:

1) $q = \min(y_i)\,; q \rightarrow H$
2) $\forall P_i \in S\ sort\ by\ \omega_i \angle(p_i; q; x)$
3) $p_i \rightarrow H$
4) Opakuj pro všechna $j < n$
5) Pokud p_j je vpravo od předešlých bodů $\rightarrow popS$
6) Pokud p_j je vlevo od předešlých bodů $p_i \rightarrow H$
7)

5 Problematické situace a jejich rozbor + ošetření těchto situací v kódu

5.1 Totožné body v množině bodů

Tento problém byl odstranění již při generování množin bodů, kde dochází ke smazání identických bodů, a vygenerování nových, aby bylo dosaženo požadované velikosti množiny.

5.1.1 Ošetření v kódu

U jednotlivých metod třídy GeneratePoints

```
    for(unsigned int j = 0; j<(pts.size()-1); j++){
        if((pts[j].x() == pts[j+1].x())&&(pts[j].y() == pts[j+1].y())){
            pts.erase(pts.begin()+j);
            j--;
        }
}
```

5.2 Body ležící v linii

Testují se 3 body konvexní obálky, pokud prostřední bod leží na linii 1. a 3. bodu, je z obáky vymazán, čímž vznikne striktně konvexní obálka.

5.2.1 Ošetření v kódu

U všech algoritmů sestavující konvexní obálku:

```
    for(int i=0; i<(ch.size()-2); i++){
        if(getPointLinePosition(ch[i+2],ch[i],ch[i+1])==ON){
            ch.remove(i+1);
            i--;
        }
}
```

6 Vstupní data, formát vstupních dat, popis

Vstupními daty je vygenerovaná množina n bodů.

V programu pomocí třídy GeneratePoints je možné vygenerovávat tyto druhy vstupních množin bodů:

Cluster – náhodně vygenerovaný hlavní bod, kolemž něho jsou vygenerovány další body, vznikají shluky bodů kde v částech množiny je mnoho bodů a v částech se body nenalézají

Random-zcela náhodně rozmístěné body, ale tím že je použita fce rand() dochází k tomu, že u velkých množin jsou body téměř rovnoměrně rozmístěny a vzhledem k tomu, že je generování nastaveno do obdélníkového okna, vzniká obdélník bodů s mnoha singulárními body na krajích

Grid – body generovány v pravidelné čtvercové síti

Circle – kruh s pevně daným středem a náhodným poloměrem

Ellipse – elipsa s pevně daným středem a náhodnými parametry a,b

StarShape – „kruh“ s pevně daným středem a pro každý bod náhodným poloměrem

Square – pevně umístěn 1 bod čtverce, náhodně dlouhá strana čtverce

7 Výstupní data, formát výstupních dat, popis

Výstupem z programu je zkonstruování a vizualizace vzniklé konvexní obálky, typ(tvar) a velikost množiny bodů n a délka trvání konstrukce konvexní obálky pro zvolený algoritmus.
Dále jsou výstupem tabulky a grafy z programu Microsoft Excel znázorňující časovou náročnost jednotlivých algoritmů na zvolených datech

7.1 Grafy ilustrující doby běhu algoritmů pro zvolená n

Pro předvedení časové náročnosti algoritmů byly zvoleny množiny bodů Cluster, Random a StarShape.

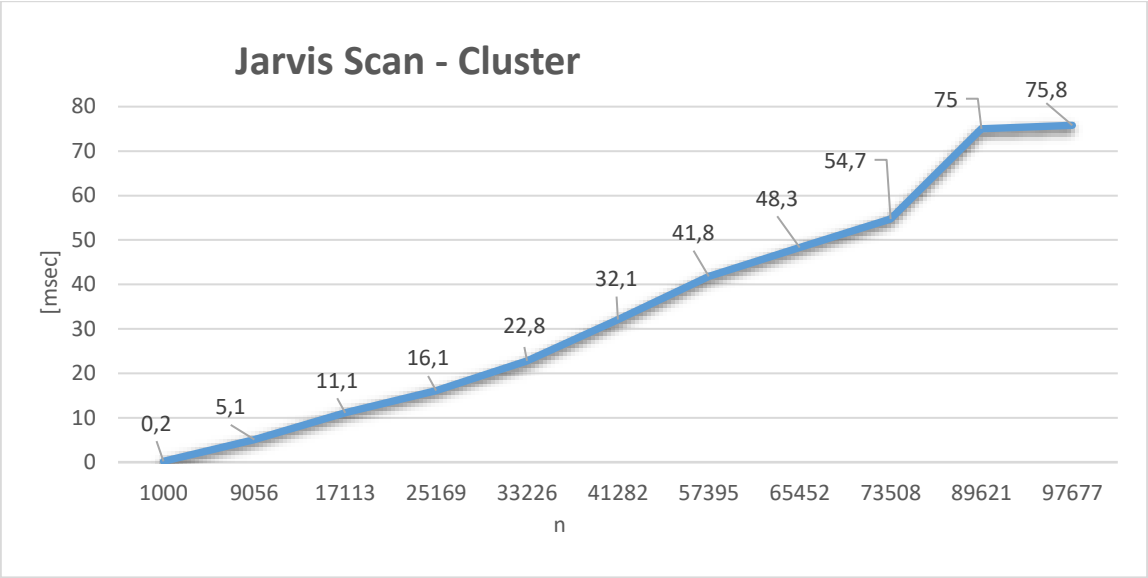
Random jak již bylo zmíněno dříve vyplňuje obdélníkové okno a svojí náročností na okrajích (mnoho bodů na linii) se blíží gridu. Grid samotný z důvodu příliš vysoké časové náročnosti zvolen nebyl.

Cluster byl zvolen, jelikož lépe vystihuje náhodný element generování bodů, jelikož body nejsou rovnoměrně rozmístěny.

StarShape byl zvolen jako ukázkový doplněk.

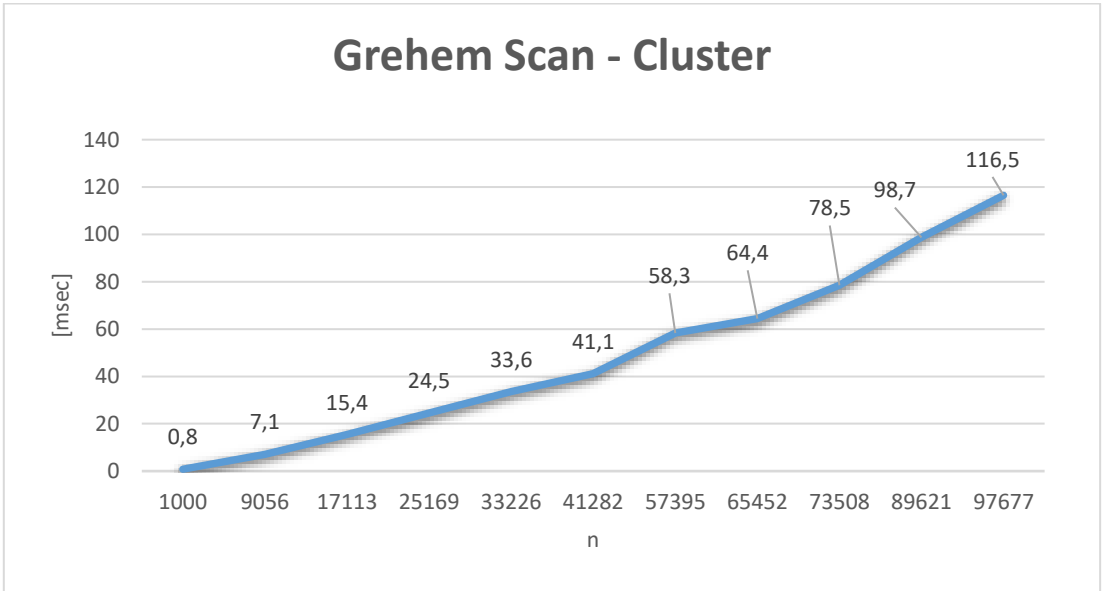
7.1.1 Cluster, Jarvis Scan

pocet bodu	1000	9056	17113	25169	33226	41282	57395	65452	73508	89621	97677
1	0	5	13	17	20	22	41	45	67	51	81
2	0	6	11	18	22	37	32	56	52	73	77
3	0	4	13	14	24	27	38	49	50	90	56
4	1	6	8	14	24	33	52	48	57	97	63
5	0	4	13	16	22	36	35	51	57	84	66
6	1	7	10	16	22	29	45	44	47	65	78
7	0	5	11	20	24	34	53	48	52	75	72
8	0	5	8	15	20	33	41	44	64	78	85
9	0	4	11	13	24	38	36	50	46	76	91
10	0	5	13	18	26	32	45	48	55	61	89
průměr	0,2	5,1	11,1	16,1	22,8	32,1	41,8	48,3	54,7	75	75,8
rozptyl	0,16	0,89	3,49	4,29	3,36	21,69	44,16	11,81	42,01	167,6	118,96



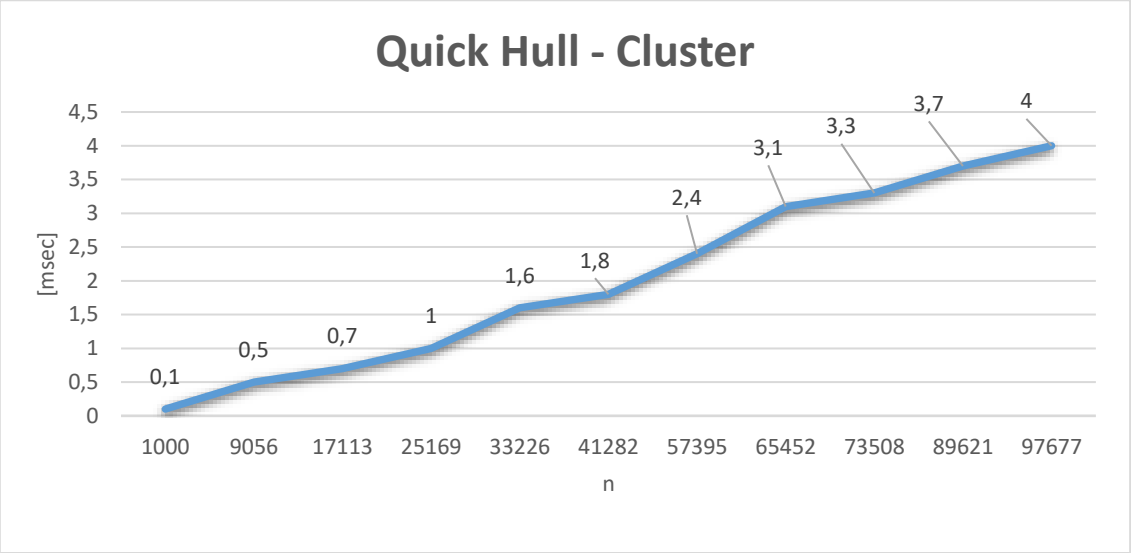
7.1.2 Cluster, Grehem Scan

pocet bodu	1000	9056	17113	25169	33226	41282	57395	65452	73508	89621	97677
1	1	7	15	22	37	38	58	63	80	94	97
2	1	6	15	25	35	44	58	64	78	99	138
3	1	7	15	27	32	39	56	68	76	87	163
4	1	7	15	27	30	41	57	63	84	93	123
5	1	7	16	24	33	42	59	64	72	100	108
6	0	8	16	25	36	40	56	62	87	94	102
7	0	7	16	23	31	41	61	64	73	109	113
8	1	8	17	24	35	45	58	71	77	100	110
9	1	7	15	23	34	39	57	63	72	122	101
10	1	7	14	25	33	42	63	62	86	89	110
průměr	0,8	7,1	15,4	24,5	33,6	41,1	58,3	64,4	78,5	98,7	116,5
rozptyl	0,16	0,29	0,64	2,45	4,44	4,49	4,41	7,44	28,45	96,01	366,65



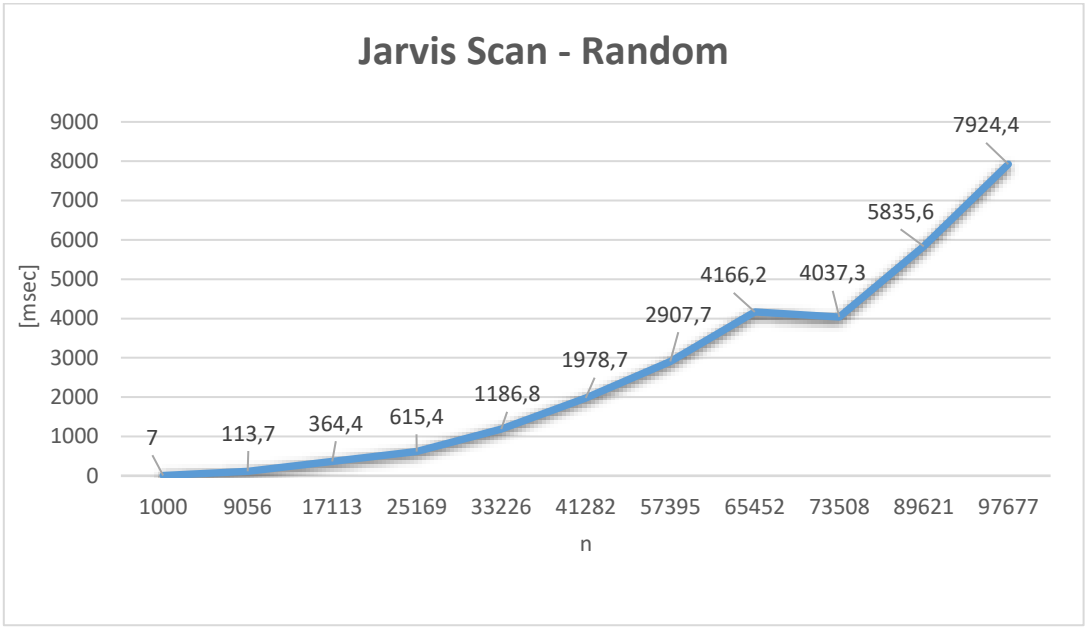
7.1.3 Cluster, Quick Hull

pocet bodu	1000	9056	17113	25169	33226	41282	57395	65452	73508	89621	97677
1	0	0	1	1	2	2	2	3	4	4	4
2	0	1	1	1	2	2	2	2	4	4	4
3	0	1	1	1	2	2	2	4	3	3	4
4	0	0	0	1	1	2	3	4	3	4	4
5	0	0	1	1	1	1	4	3	3	4	4
6	0	0	1	1	2	2	2	2	4	3	4
7	0	1	1	1	1	2	2	3	3	3	4
8	1	0	0	1	2	1	3	3	4	4	4
9	0	1	1	1	2	2	2	3	2	4	4
10	0	1	0	1	1	2	2	4	3	4	4
průměr	0,1	0,5	0,7	1	1,6	1,8	2,4	3,1	3,3	3,7	4
rozptyl	0,09	0,25	0,21	0	0,24	0,16	0,44	0,49	0,41	0,21	0



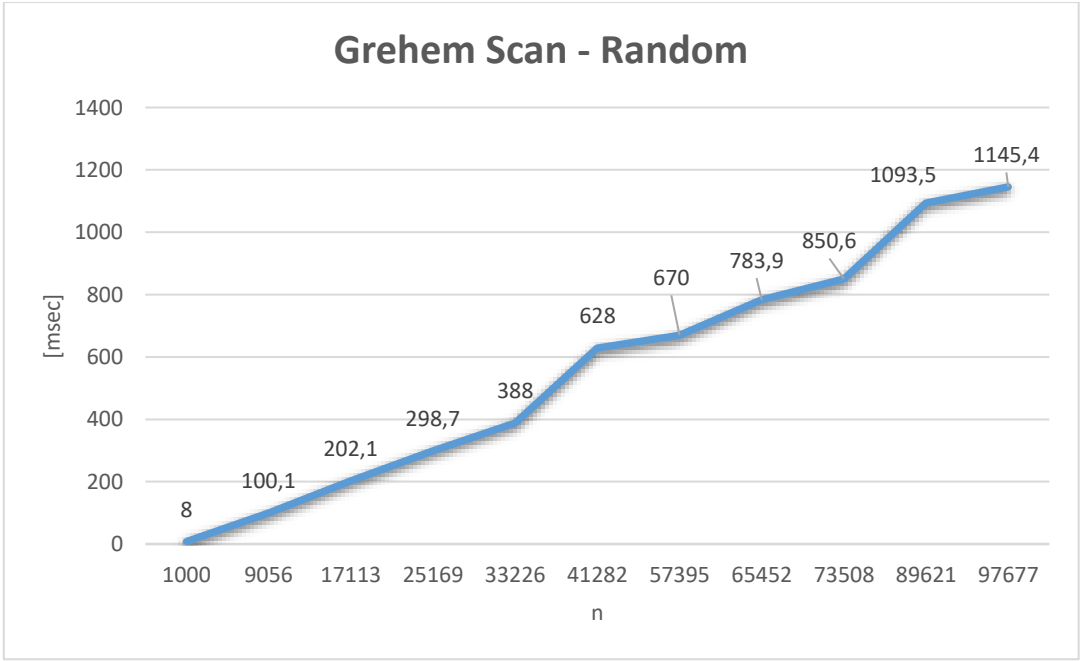
7.1.4 Random, Jarvis Scan

pocet bodu	1000	9056	17113	25169	33226	41282	57395	65452	73508	89621	97677
1	8	128	491	497	855	2606	1862	4169	3874	4726	8410
2	8	80	234	436	880	1577	1955	3878	2886	7241	5858
3	6	101	206	778	1398	2534	2295	4731	5285	4278	8451
4	8	76	275	525	1462	1410	2045	2461	3169	4087	8540
5	7	116	447	459	1230	1432	3375	4468	3240	6781	3979
6	6	144	400	1004	1394	1938	3456	4396	4705	4066	9264
7	6	128	318	466	881	2482	3810	4136	2692	4368	9709
8	8	124	409	584	1632	1803	3172	4455	5529	8176	8412
9	7	122	486	744	1470	2431	3197	4571	3287	7396	8623
10	6	118	378	661	666	1574	3910	4397	5706	7237	7998
průměr	7	113,7	364,4	615,4	1186,8	1978,7	2907,7	4166,2	4037,3	5835,6	7924,4
rozptyl	0,8	424,41	9331,84	29820,84	101250,76	213972,21	561208,01	375511,36	1212688,01	2474543,84	2643880,64



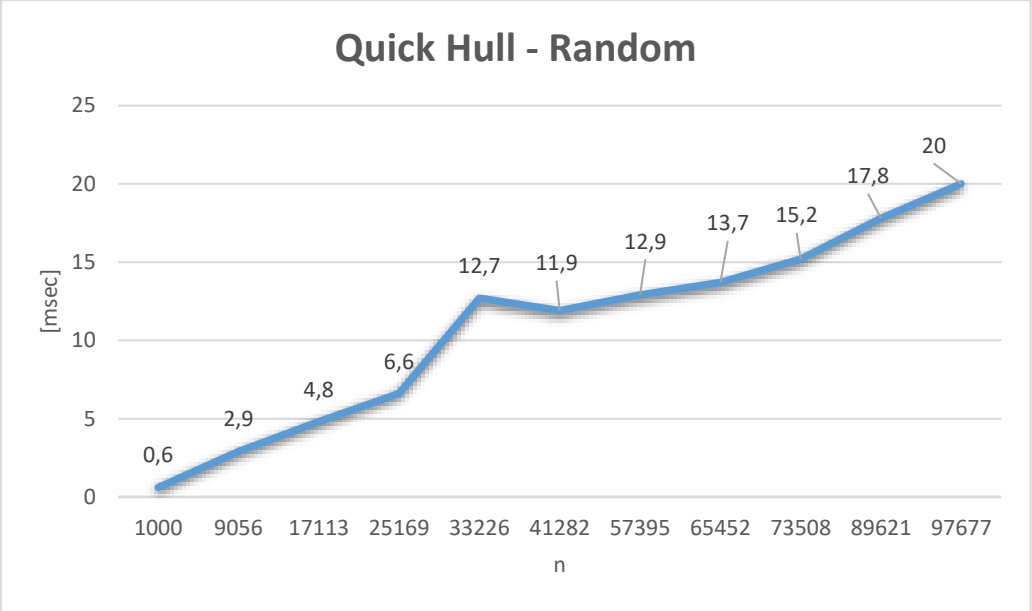
7.1.5 Random, Grehem Scan

pocet bodu	1000	9056	17113	25169	33226	41282	57395	65452	73508	89621	97677
1	8	99	180	277	411	728	679	754	898	1303	1126
2	8	100	216	306	423	732	647	811	884	1051	1093
3	9	104	201	323	416	723	654	760	859	1027	1169
4	8	97	224	274	391	717	644	764	876	1082	1175
5	8	99	202	310	353	708	4332	773	813	1000	1101
6	7	101	217	273	370	488	695	898	828	968	1199
7	8	99	179	314	2127	489	682	803	852	1250	1098
8	9	104	196	290	388	2238	663	791	812	1100	1200
9	8	96	203	304	376	506	624	715	848	1053	1135
10	7	102	203	316	364	561	742	770	836	1101	1158
průměr	8	100,1	202,1	298,7	388	628	670	783,9	850,6	1093,5	1145,4
rozptyl	0,4	6,49	195,69	315,01	272652,49	243534,6	1207887,96	2108,89	771,44	10097,45	1489,44



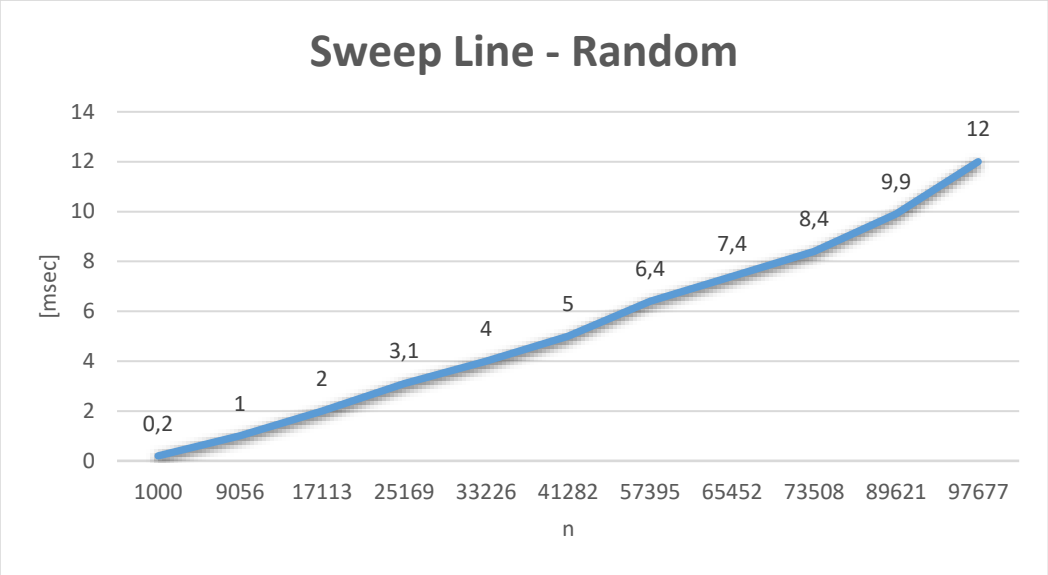
7.1.6 Random, Quick Hull

pocet bodu	1000	9056	17113	25169	33226	41282	57395	65452	73508	89621	97677
1	1	3	5	6	14	11	13	13	14	18	16
2	0	3	5	8	13	13	13	13	15	19	20
3	1	3	5	7	13	12	13	14	15	20	21
4	0	3	4	6	12	12	14	11	16	14	19
5	1	4	5	7	12	13	12	16	15	18	18
6	1	2	5	7	11	11	15	14	15	17	23
7	1	3	5	6	13	12	13	14	17	19	22
8	0	2	5	6	13	13	10	16	13	18	20
9	0	3	4	7	14	11	13	13	16	17	20
10	1	3	5	6	12	11	13	13	16	18	21
průměr	0,6	2,9	4,8	6,6	12,7	11,9	12,9	13,7	15,2	17,8	20
rozptyl	0,24	0,29	0,16	0,44	0,81	0,69	1,49	2,01	1,16	2,36	3,6



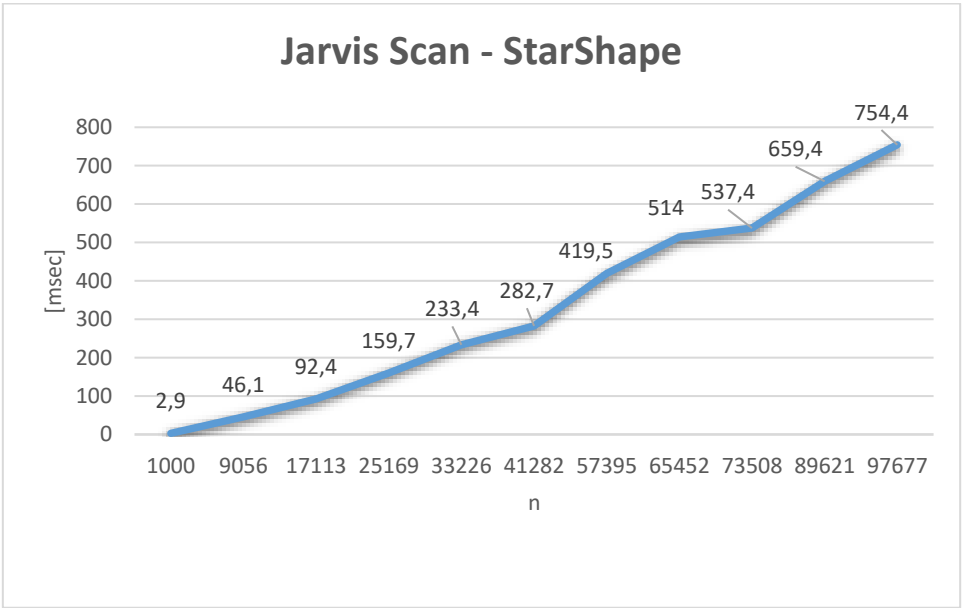
7.1.7 Random, Sweep Line

pocet bodu	1000	9056	17113	25169	33226	41282	57395	65452	73508	89621	97677
1	0	1	1	3	4	5	6	8	8	10	11
2	0	1	2	4	4	5	7	7	10	9	13
3	0	1	1	3	3	4	7	8	9	10	12
4	0	1	2	4	4	5	6	7	7	10	12
5	0	1	3	2	4	5	6	7	8	10	13
6	1	1	3	2	4	5	7	8	8	10	12
7	0	1	2	3	4	5	7	7	8	11	13
8	0	1	2	4	4	6	6	7	9	9	12
9	1	1	2	3	4	6	6	7	8	10	11
10	0	1	2	3	5	4	6	8	9	10	11
průměr	0,2	1	2	3,1	4	5	6,4	7,4	8,4	9,9	12
rozptyl	0,16	0	0,4	0,49	0,2	0,4	0,24	0,24	0,64	0,29	0,6



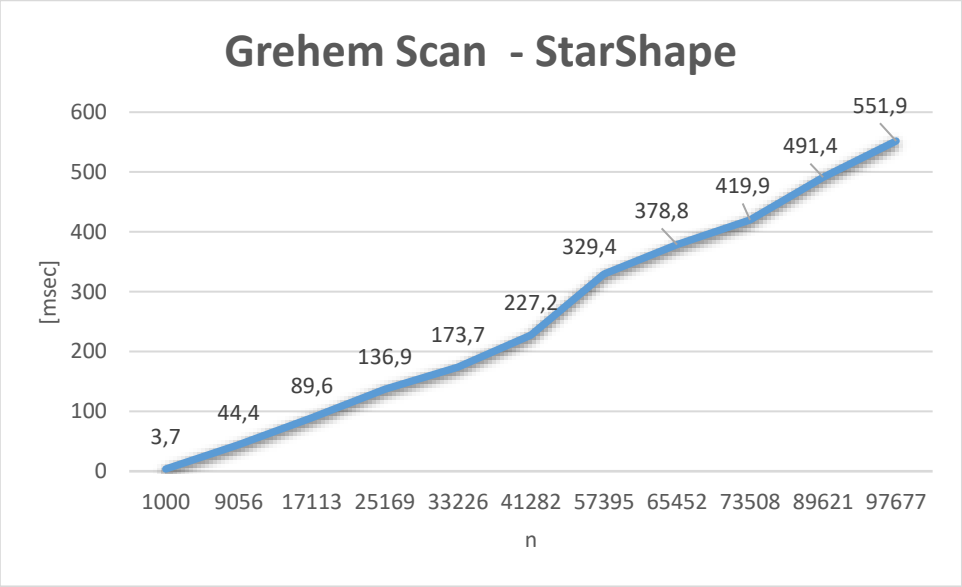
7.1.8 StarShape, Jarvis Scan

pocet bodu	1000	9056	17113	25169	33226	41282	57395	65452	73508	89621	97677
1	3	45	92	152	253	312	415	527	525	630	702
2	3	42	95	151	240	330	398	525	530	728	754
3	3	45	79	131	232	258	387	534	572	608	849
4	3	48	97	178	250	276	420	483	522	597	845
5	3	47	89	172	241	295	496	500	526	701	645
6	3	46	102	156	213	234	405	442	545	741	871
7	4	52	98	172	201	277	462	520	571	635	688
8	2	43	96	181	286	286	440	517	519	619	666
9	3	52	90	158	213	278	366	525	550	641	770
10	2	41	86	146	205	281	406	567	514	694	754
průměr	2,9	46,1	92,4	159,7	233,4	282,7	419,5	514	537,4	659,4	754,4
rozptyl	0,29	12,89	40,24	225,41	621,84	634,21	1287,25	1000,6	400,44	2425,84	5793,44



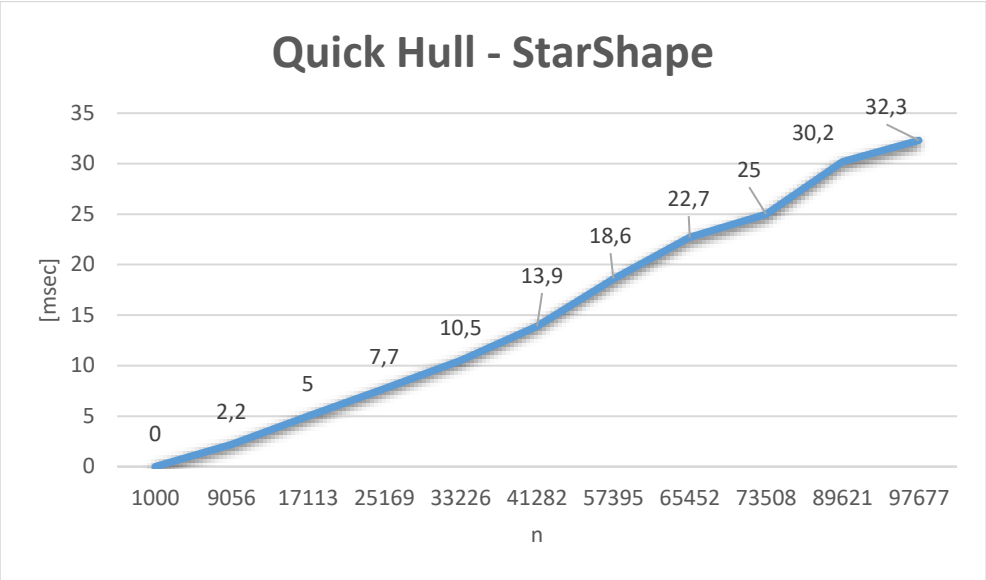
7.1.9 StarShape, Grehem Scan

pocet bodu	1000	9056	17113	25169	33226	41282	57395	65452	73508	89621	97677
1	4	43	86	148	170	237	292	410	415	488	493
2	3	42	87	156	174	211	345	378	406	502	558
3	5	44	89	125	183	215	339	367	402	457	576
4	4	42	86	139	173	217	330	374	416	500	556
5	4	44	85	126	171	224	333	373	413	483	585
6	4	43	85	128	171	258	345	389	430	490	545
7	3	51	88	135	172	228	298	386	419	498	517
8	3	43	87	135	175	213	325	380	428	520	574
9	4	48	109	128	176	232	345	382	424	487	524
10	3	44	94	149	172	237	342	349	446	489	591
průměr	3,7	44,4	89,6	136,9	173,7	227,2	329,4	378,8	419,9	491,4	551,9
rozptyl	0,41	7,44	48,04	106,49	12,81	189,16	339,84	222,56	146,69	234,04	926,09



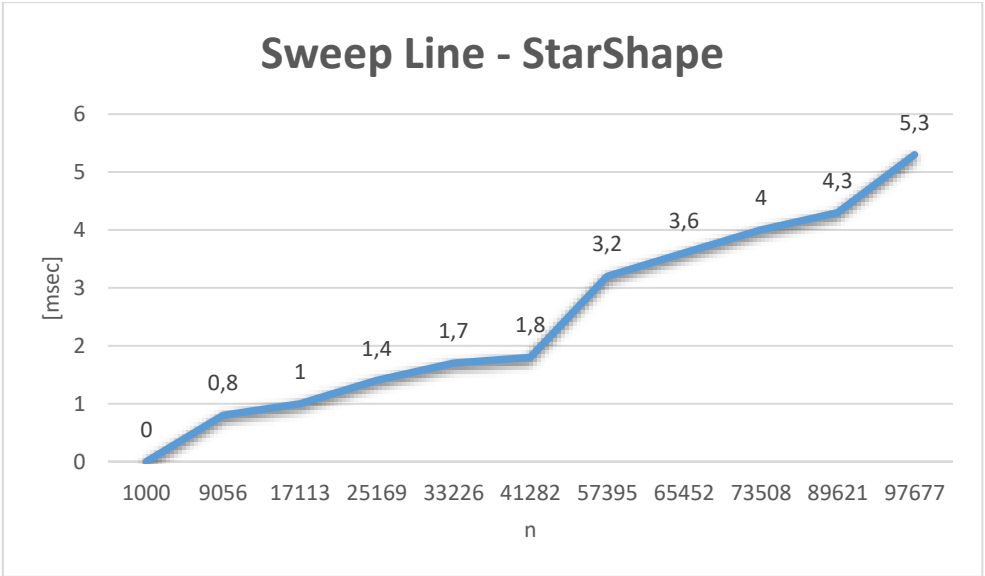
7.1.10 StarShape, Quick Hull

pocet bodu	1000	9056	17113	25169	33226	41282	57395	65452	73508	89621	97677
1	0	2	5	8	10	13	19	20	26	29	30
2	0	2	5	8	10	15	19	20	25	31	32
3	0	3	5	7	9	14	17	24	22	32	32
4	0	2	5	7	11	12	18	23	25	30	30
5	0	2	5	7	12	13	19	24	25	31	37
6	0	2	6	7	10	14	19	22	29	31	34
7	0	2	5	7	10	13	19	23	25	27	34
8	0	2	5	9	10	15	21	24	23	36	30
9	0	2	4	9	11	15	17	21	27	27	31
10	0	3	5	8	12	15	18	26	23	28	33
průměr	0	2,2	5	7,7	10,5	13,9	18,6	22,7	25	30,2	32,3
rozptyl	0	0,16	0,2	0,61	0,85	1,09	1,24	3,41	3,8	6,56	4,61

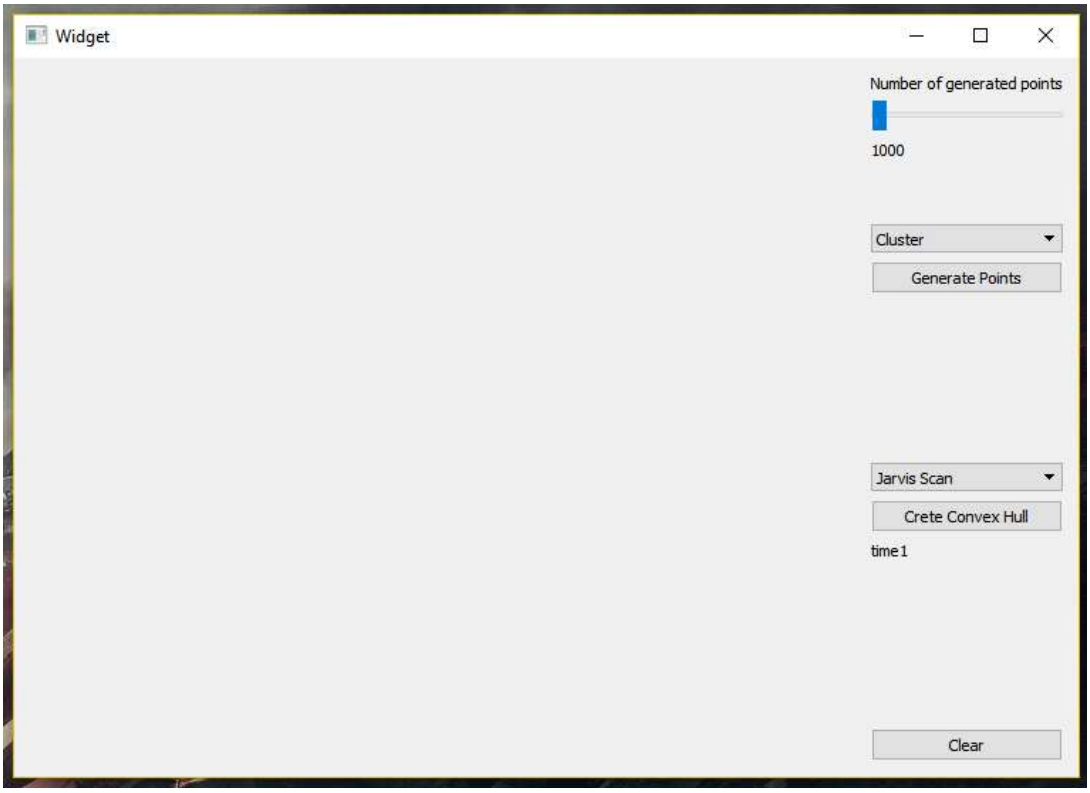


7.1.11 StarShape, Sweep Line

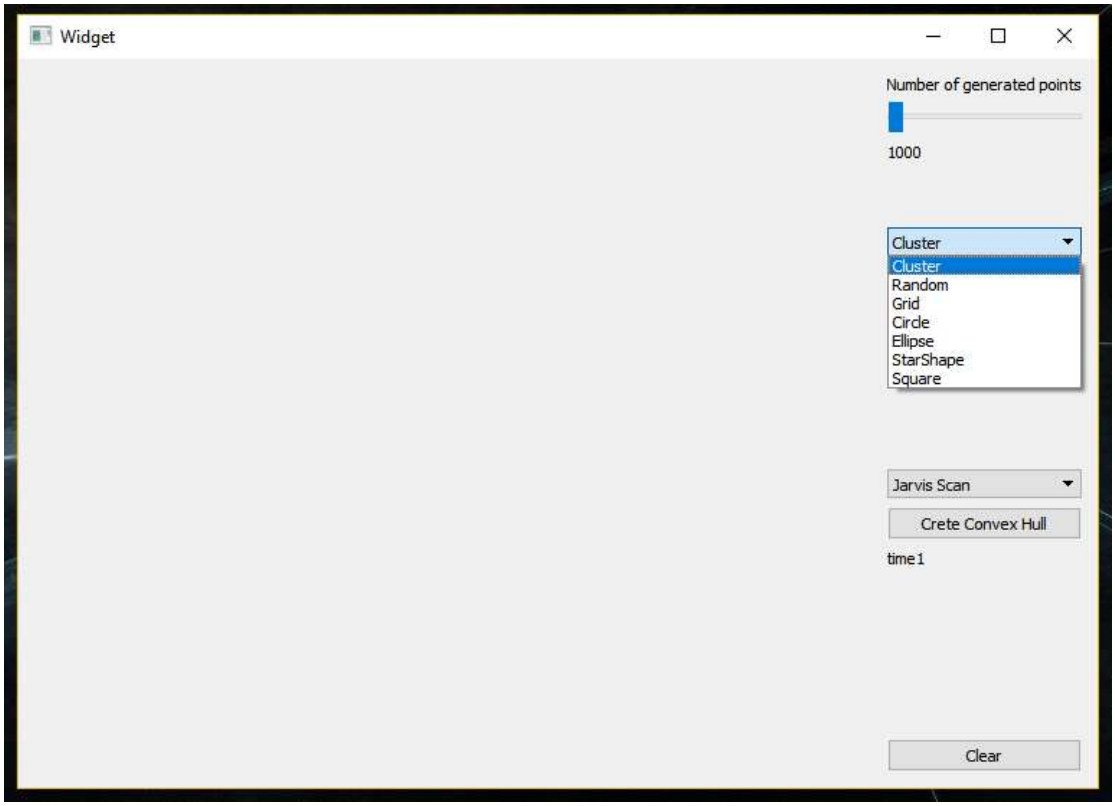
pocet bodu	1000	9056	17113	25169	33226	41282	57395	65452	73508	89621	97677
1	0	1	1	1	2	2	3	3	4	5	4
2	0	1	1	2	2	2	4	3	4	4	5
3	0	0	1	2	2	2	4	4	3	5	5
4	0	1	1	1	1	0	2	4	5	4	5
5	0	0	1	1	2	2	3	4	4	5	6
6	0	1	1	1	2	2	3	4	4	4	5
7	0	1	1	2	2	2	3	3	4	4	6
8	0	1	1	1	1	2	3	4	4	4	6
9	0	1	1	2	2	2	3	4	4	4	6
10	0	1	1	1	1	2	4	3	4	4	5
průměr	0	0,8	1	1,4	1,7	1,8	3,2	3,6	4	4,3	5,3
rozptyl	0	0,16	0	0,24	0,21	0,36	0,36	0,24	0,2	0,21	0,41



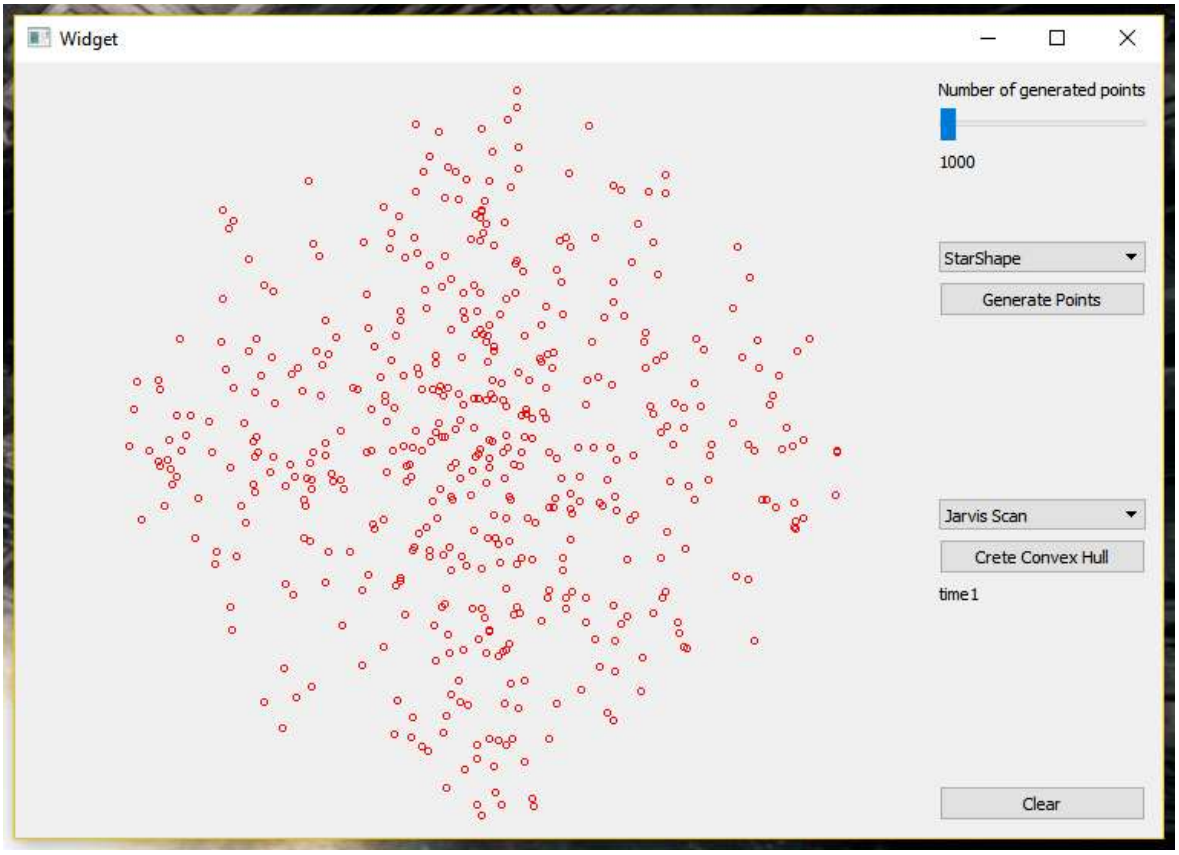
8 Printscreen vytvořené aplikace



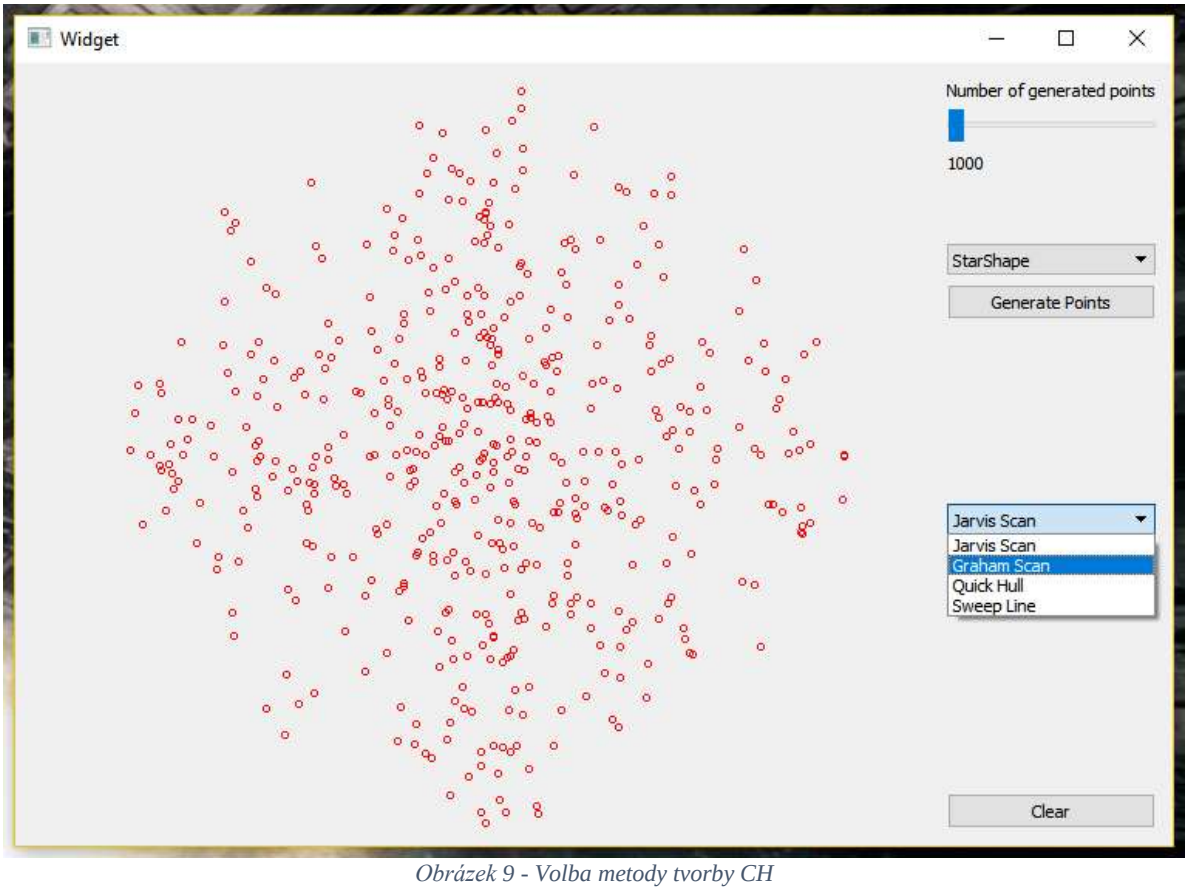
Obrázek 6 - Idle program



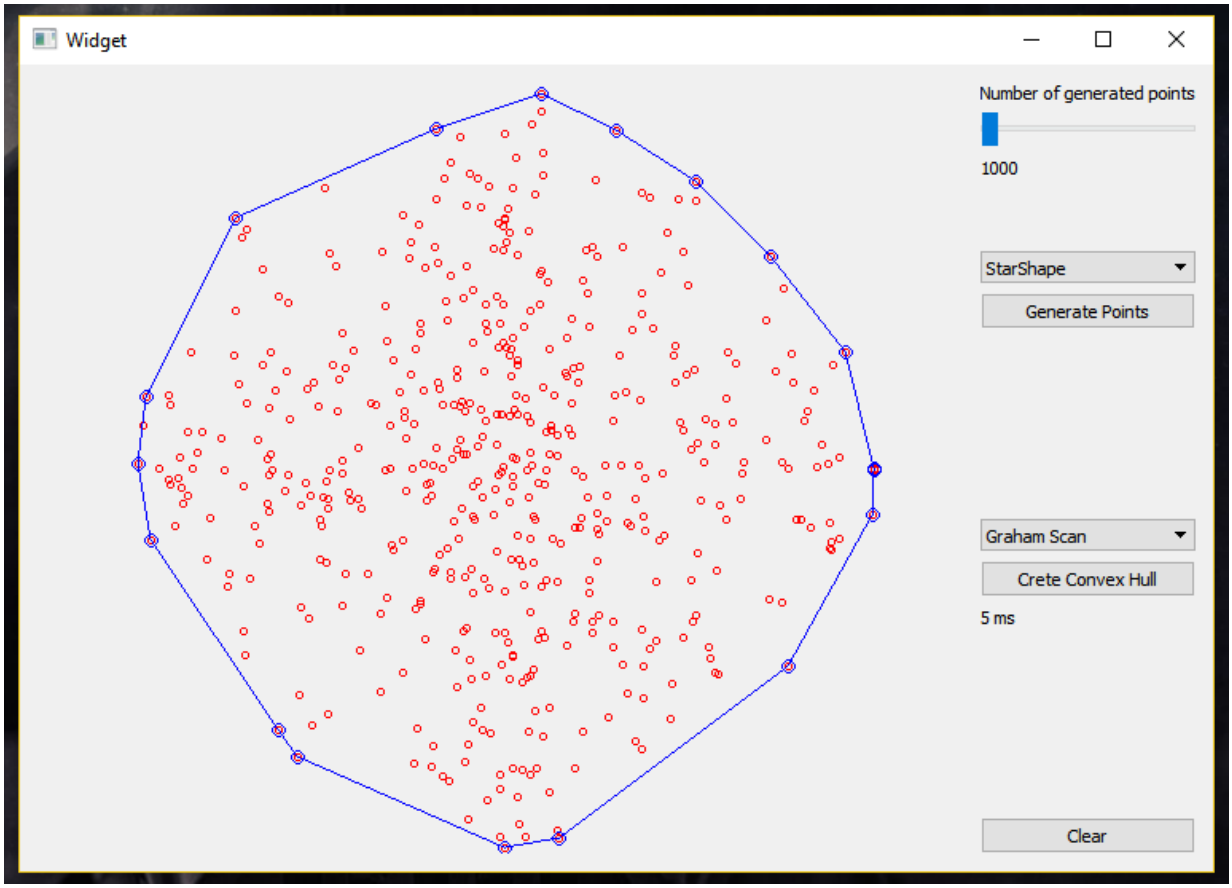
Obrázek 7 - Možnosti generování množiny bodů



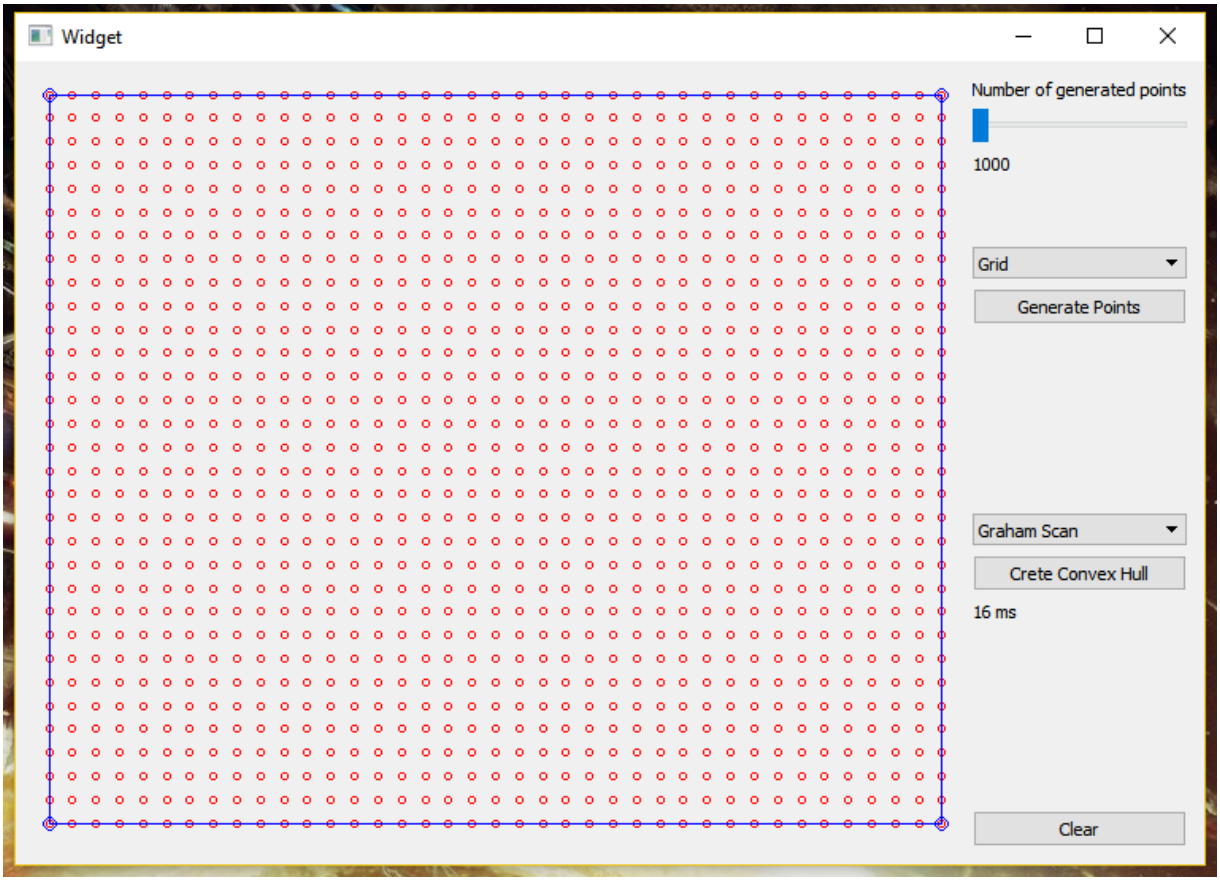
Obrázek 8 - Vygenerované body, star shape



Obrázek 9 - Volba metody tvorby CH



Obrázek 10 - CH, body CH vyznačeny modře



Obrázek 11 - Ukázka striktně konvexní obálky

9 Dokumentaci: popis tříd, datových položek a jednotlivých metod

9.1 Třída Algorithms

Třída obsahující veškeré početní výkony pro vytváření konvexní obálky

9.1.1 Metody třídy Algorithms

```
static TPosition getPointLinePosition(QPoint &q, QPoint &a, QPoint &b);
    vstupem bod a koncové body úsečky
    výstupem poloha bodu k úsečce

static double get2LinesAngle(QPoint &p1,QPoint &p2,QPoint &p3, QPoint &p4);
    vstupem koncové body 2 úseček
    výstupem úhel mezi úsečkami

static double getPointLineDistance(QPoint &q, QPoint &a, QPoint &b);
    vstupem bod a koncové body úsečky
    výstupem vzdálenost bodu od úsečky

static void qh (int s, int e, vector<QPoint> &p, QPolygon &h);
    část výpočtu QHULL pro propojení horní a dolní části

static QPolygon CHJarvis (vector<QPoint> &points);
static QPolygon CHGraham (vector<QPoint> &points);
static QPolygon QHull (vector<QPoint> &points);
static QPolygon CHSweep (vector<QPoint> &points);
    vstupem množina bodů
    výstupem konvexní obálka množiny

static double distance(QPoint a, QPoint b);
    vstup 2 body
    výstup délka mezi body
```

9.2 Třída Draw

Slouží k vykreslení množiny a obálky

9.2.1 Datové položky třídy Draw

```
std::vector<QPoint> points;        množina bodů
QPolygon ch;                       konvexni obalka
```

9.2.2 Metody třídy Draw

```
void paintEvent(QPaintEvent *e);
    metoda k provedení vykreslování

void mousePressEvent(QMouseEvent *e);
    vykreslení bodu po kliknutí

void setCH( QPolygon ch_) {ch = ch_;}
    předání konvexní obálky

void setPoints(std::vector<QPoint> pts){points = pts;}
    předání množiny bodů

std::vector<QPoint> getPoints(){return points;}
    přidání nakliknutého bodu do množiny

void clearAll();
    smaže vše

void clearHull();
        smaže jen konvexní obálku
```

9.3 Třída Widget

9.3.1 Sloty třídy Widget

```
void on_pushButton_2_clicked();
    smaže vše a nastaví defaultní hodnoty

void on_points_clicked();
    vygeneruje množinu bodů

void on_horizontalSlider_sliderMoved(int position);
    po pohnutí posuvníkem nastaví hodnotu n

void on_pushButton_3_clicked();
        vytvoří konvexní obálku
```

9.4 Třída GeneratePoints

Slouží k generování množin bodů

9.4.1 Metody třídy GeneratePoints

```
static std::vector<QPoint> generateCluster(int &n,QSize &size);
static std::vector<QPoint> generateRandom(int &n,QSize &size);
static std::vector<QPoint> generateGrid(int &n,QSize &size);
static std::vector<QPoint> generateCircle(int &n, QSize &size);
static std::vector<QPoint> generateEllipse(int &n, QSize &size);
static std::vector<QPoint> generateStarShape(int &n, QSize &size);
static std::vector<QPoint> generateSquare(int &n, QSize &size);
        vstupem je velikost množiny bodů a velikost okna kde se mají vygenerovat
        výstupem je množina bodů daného tvaru
```

10 Závěr, možné či neřešené problémy, náměty na vylepšení

10.1 Závěr

V rámci úlohy byla vytvořena aplikace, která dokáže generovat množiny bodů (cluster, random, grid, circle, ellipse, starshape, square) a nad těmito množinami bodů dokáže zkonstruovat konvexní obálku pomocí algoritmů Jarvis Scan,, Graham Scan, Quick Hull a Sweep Line. V kódu bylo ošetřeno, aby zkonstruované obálky byly striktně konvexní a aby při generování bodů nedocházelo ke generování duplicitních bodů. Program po sestrojení konvexní obálky ukáže časovou náročnost výpočtu v [ms].

10.2 Náměty na vylepšení

Po porovnání s ostatními programátory bylo zjištěno že časová náročnost algoritmů je mnohem větší než u ostatních, bylo by dobré zjistit, zda je to pouze způsobeno stářím počítače, na kterém bylo testováno, nebo se jedná o chybu v kódu.

10.3 Neřešené problémy

Během měření časové náročnosti jednotlivých algoritmů bylo zjištěno, že v programu u některých kombinací množina/algoritmus padá s hláškou:

terminate called after throwing an instance of 'std::bad_alloc'
what(): std::bad_alloc

Zhotovitel této aplikace však prohlašuje, že nemá sebemenší tušení, jak to napravit.