

Zadání projektu 2 - Práce s datovými strukturami

Popis projektu

Vytvořte program, který implementuje jednoduchou shlukovou analýzu, metodu nejbližšího souseda (angl. single linkage).

Smyslem projektu není studium shlukových analýz. Pro projekt bude stačit následující popis (zdroj [Wikipedia](#)): *Shluková analýza (též clusterová analýza, anglicky cluster analysis) je vícerozměrná statistická metoda, která se používá ke klasifikaci objektů. Slouží k třídění jednotek do skupin (shluků) tak, že jednotky náležící do stejné skupiny jsou si podobnější než objekty z ostatních skupin.*

Shlukovou analýzu provádějte na dvourozměrných objektech. Každý objekt je identifikován celým číslem. Objekty jsou uloženy v textovém souboru.

Při implementaci můžete pro vizualizaci a porozumění objektů používat [tuto jednoduchou aplikaci](#), která vykresluje a obarvuje vámi vygenerované shluky.

Metoda nejbližšího souseda

Metoda nejbližšího souseda vybírá ke spojení vždy dva shluky, které mají k sobě nejblíže. Vzdálenost dvou shluků je rovna nejmenší vzdálenosti libovolných dvou objektů z obou shluků.

Detailní specifikace

Překlad a odevzdání zdrojového souboru

Odevzdání: Program implementujte ve zdrojovém souboru `cluster.c`. Zdrojový soubor odevzdejte prostřednictvím informačního systému.

Překlad: Program bude překládán s následujícími argumenty

```
$ gcc -std=c99 -Wall -Wextra -Werror -DNDEBUG cluster.c -o cluster -lm
```

- Definice makra `NDEBUG` (argument `-DNDEBUG`) je z důvodu anulování efektu ladicích informací.
- Propojení s matematickou knihovnou (argument `-lm`) je z důvodu výpočtu vzdálenosti objektů.

Syntax spuštění

Program se spouští v následující podobě:

```
./cluster SOUBOR [N]
```

Argumenty programu:

- `SOUBOR` je jméno souboru se vstupními daty.

- `N` je volitelný argument definující cílový počet shluků. $N > 0$. Výchozí hodnota (při absenci argumentu) je 1.

Implementační detaily

Formát vstupního souboru

Vstupní data jsou uložena v textovém souboru. První řádek souboru je vyhrazen pro počet objektů v souboru a má formát:

`count=N`

kde číslo je počet objektů v souboru. Následuje na každém řádku definice jednoho objektu. Počet řádků souboru odpovídá minimálně počtu objektů + 1 (první řádek). Další řádky souboru ignorujte. Řádek definující objekt je formátu:

`OBJID X Y`

kde OBJID je v rámci souboru jednoznačný celočíselný identifikátor, X a Y jsou souřadnice objektu také celá čísla. Platí $0 \leq X \leq 1000$, $0 \leq Y \leq 1000$.

1. podúkol

Stáhněte si kostru programu [cluster.c](#). Seznamte se s datovými typy a funkcemi. Vaším úkolem je pouze doplnit kód na místech označených komentářem **TODO**.

2. podúkol

Načítání vstupního souboru a následný výpis:

1. Implementujte funkce:

```
void init_cluster(struct cluster_t *c, int cap);
```

```
void clear_cluster(struct cluster_t *c);
```

```
void append_cluster(struct cluster_t *c, struct obj_t obj);
```

```
int load_clusters(char *filename, struct cluster_t **arr);
```

Funkce `init_cluster` slouží pro inicializaci shluku (alokaci požadovaného místa).

Funkce `clear_cluster` slouží pro odstranění všech objektů ve shluku (dealokaci místa) a reinicializaci shluku s kapacitou 0.

Funkce `append_cluster` slouží pro přidání objektu na konec shluku.

Funkce `load_clusters` načítá ze vstupního souboru všechny objekty a ukládá je každý do jednoho shluku. Shluky budou uloženy v poli. Místo pro pole shluků musí funkce alokovat.

2. Ověřte funkcionalitu na načtení vstupního souboru (pomocí vámi implementované funkce `load_clusters`) a následném výpisu (pomocí funkce `print_clusters`):

Vstupní soubor `objekty`

`count=20`

40 86 663
43 747 938
47 285 973
49 548 422
52 741 541
56 44 854
57 795 59
61 267 375
62 85 874
66 125 211
68 80 770
72 277 272
74 222 444
75 28 603
79 926 463
83 603 68
86 238 650
87 149 304
89 749 190
93 944 835

Načtení vstupního souboru a následné vypsání shluků:

\$./cluster objekty 20

Clusters:

cluster 0: 40[86,663]
cluster 1: 43[747,938]
cluster 2: 47[285,973]
cluster 3: 49[548,422]
cluster 4: 52[741,541]
cluster 5: 56[44,854]
cluster 6: 57[795,59]
cluster 7: 61[267,375]
cluster 8: 62[85,874]
cluster 9: 66[125,211]
cluster 10: 68[80,770]
cluster 11: 72[277,272]
cluster 12: 74[222,444]
cluster 13: 75[28,603]
cluster 14: 79[926,463]
cluster 15: 83[603,68]

cluster 16: 86[238,650]

cluster 17: 87[149,304]

cluster 18: 89[749,190]

cluster 19: 93[944,835]

3. podúkol

Implementujte všechny ostatní funkce v kostře souboru cluster.c označené komentářem TODO. Výsledný program odevzdejte.

Příklady vstupů a výstupů

```
$ cat objekty
```

```
count=20
```

```
40 86 663
```

```
43 747 938
```

```
47 285 973
```

```
49 548 422
```

```
52 741 541
```

```
56 44 854
```

```
57 795 59
```

```
61 267 375
```

```
62 85 874
```

```
66 125 211
```

```
68 80 770
```

```
72 277 272
```

```
74 222 444
```

```
75 28 603
```

```
79 926 463
```

```
83 603 68
```

```
86 238 650
```

```
87 149 304
```

```
89 749 190
```

```
93 944 835
```

```
$ ./cluster objekty 8
```

```
Clusters:
```

```
cluster 0: 40[86,663] 56[44,854] 62[85,874] 68[80,770] 75[28,603] 86[238,650]
```

```
cluster 1: 43[747,938]
```

```
cluster 2: 47[285,973]
```

```
cluster 3: 49[548,422]
```

```
cluster 4: 52[741,541] 79[926,463]
```

```
cluster 5: 57[795,59] 83[603,68] 89[749,190]
```

```

cluster 6: 61[267,375] 66[125,211] 72[277,272] 74[222,444] 87[149,304]
cluster 7: 93[944,835]
$ valgrind ./cluster objekty 8
==23223== Memcheck, a memory error detector
==23223== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==23223== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==23223== Command: ./cluster objekty 8
==23223==
Clusters:
cluster 0: 40[86,663] 56[44,854] 62[85,874] 68[80,770] 75[28,603] 86[238,650]
cluster 1: 43[747,938]
cluster 2: 47[285,973]
cluster 3: 49[548,422]
cluster 4: 52[741,541] 79[926,463]
cluster 5: 57[795,59] 83[603,68] 89[749,190]
cluster 6: 61[267,375] 66[125,211] 72[277,272] 74[222,444] 87[149,304]
cluster 7: 93[944,835]
==23223==
==23223== HEAP SUMMARY:
==23223==    in use at exit: 0 bytes in 0 blocks
==23223== total heap usage: 36 allocs, 36 frees, 9,748 bytes allocated
==23223==
==23223== All heap blocks were freed -- no leaks are possible
==23223==
==23223== For counts of detected and suppressed errors, rerun with: -v
==23223== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

```

Hodnocení

Na výsledném hodnocení mají hlavní vliv následující faktory:

- přesné dodržení implementačních detailů,
- implementace jednotlivých funkcí,
- správná práce s pamětí,
- správný algoritmus shlukování,
- správné řešení neočekávaných stavů.