**UNIVERZITA SV. CYRILA A METODA V TRNAVE**

**Shape

Description automatically generated with medium confidenceFAKULTA PRÍRODNÝCH VIED**

**Zadanie č. 13**

**Letná verzia**

**Semestrálna práca** 26.5.2022 Pavol Lukačka, APIN-BcD19

**Popis problému**

Riešte problém, ktorý v súbore nájde prvky (slová), ktorých dĺžka je N znakov (zadané užívateľom), vypíšte ich na obrazovku aj do druhého súboru, zoradené podľa abecedy a vypíše ich počet.

**Obsah:**

[1. Letná analýza problému 7](#_Toc104435147)

[1.1 Načítanie a vpisovanie do súboru a celkovo problematika vstupu a výstupu 7](#_Toc104435148)

[1.1.1 Problematika alokácie 8](#_Toc104435149)

[1.1.2 Dynamická alokácia 9](#_Toc104435150)

[1.1.3 Výhody dynamickej alokácie: 10](#_Toc104435151)

[1.1.4 Nevýhody dynamickej alokácie: 10](#_Toc104435152)

[1.1.6 Prečo som neimplementoval binárny súbor? 12](#_Toc104435153)

[1.2 problém definícia „slov“ a ich selekcia 12](#_Toc104435154)

[1.3 problém abecedné zoradenie & sortovací algoritmus 13](#_Toc104435155)

[1.3.1 Aký sortovací algoritmus som použil v zime? 13](#_Toc104435156)

[1.3.2 QuickSort a Radix Sort v komplexite 14](#_Toc104435157)

[1.3.3 Rozdelenie Radixsortu na LSD (Least Significant digit) A MSD (Most significant digit) 15](#_Toc104435158)

[1.3.4 Radixsort alebo Quicksort? 16](#_Toc104435159)

[1.3.5 Ako som sa výsledne rozhodol? 17](#_Toc104435160)

[1.3.6 Bubblesort LETO v mojom programe 18](#_Toc104435161)

[1.3.7 Quicksort v mojom programe 19](#_Toc104435162)

[1.3.8 Krokovanie Quicksortu. 20](#_Toc104435163)

[1.3.9 Prečo quicksort dáva nesprávne honoty pri velkom počte znakov? 22](#_Toc104435164)

[1.3.5 Radixsort v mojom programe 23](#_Toc104435165)

[3.3.6 Vysvetelnie Radixsortu. 24](#_Toc104435166)

[1.4 problém výskyt slov 25](#_Toc104435167)

[2. Opravené chyby v programe 26](#_Toc104435168)

[2.1 CHYBA C. 1 - Duplikácia slov vo výpise 27](#_Toc104435169)

[2.2 CHYBA 2 – VYMAZ WHITESPACES - DUPLIKOVANE VOLANIE 29](#_Toc104435170)

[2.3 CHYBA 3 – POCITADLO MEDZIER - DUPLIKOVANE VOLANIE 30](#_Toc104435171)

[2.4 Test volaní všetkých funkcií: 32](#_Toc104435172)

[3. MERANIA 34](#_Toc104435173)

[3.1 Príklad výpisu meraní: 35](#_Toc104435174)

[3.1.1 ZIMA - BUBBLE SORT - Testovací retazec 3 : 35](#_Toc104435175)

[3.1.2 LETO BUBBLE SORT - Testovací retazec 3: 35](#_Toc104435176)

[3.1.3 LETO QUICK SORT - Testovací retazec 3: 35](#_Toc104435177)

[3.1.4 LETO RADIX SORT - Testovací retazec 3: 35](#_Toc104435178)

[3.1 Testovacie pomocou duplicitného opakovania funkcií 36](#_Toc104435179)

[3.1.1 ZIMA - BUBBLE SORT - Testovací retazec 1, 2, 3 : 37](#_Toc104435180)

[3.1.2 LETO BUBBLE SORT - Testovací retazec 1, 2, 3 : 38](#_Toc104435181)

[3.1.3 LETO QUICK SORT Testovací retazec 1,2,3 : 39](#_Toc104435182)

[3.1.4 LETO RADIX SORT - Testovací retazec 1,2,3 40](#_Toc104435183)

[3.2 Spriemerované Merania samotného programu 41](#_Toc104435184)

[4. Vývojové diagramy 42](#_Toc104435185)

[4.1 void vymaz\_extra\_whitespaces(char\* pole\_char) 42](#_Toc104435186)

[4.2 FLOWCHART pole\_org 43](#_Toc104435187)

[4.3 FLOWCHART Bubble sort 44](#_Toc104435188)

[4.4 FLOWCHART quickSort 45](#_Toc104435189)

[4.5 radixSort 46](#_Toc104435190)

[4.6 FLOWCHART vyskyt 47](#_Toc104435191)

[4.7 FLOWCHART vyskyt\_vypis 48](#_Toc104435192)

[4.8 FLOWCHART switch\_volba 49](#_Toc104435193)

[4.9 FLOWCHART void export\_novy\_retazec 50](#_Toc104435194)

[4.10 FLOWCHART nahr\_interpunkcie 51](#_Toc104435195)

[5. Definicia vstupných,pomocných a výstupných premenných: 52](#_Toc104435196)

[5.1 void vymaz\_extra\_whitespaces - definicia vstupných,pomocných a výstupných premenných 52](#_Toc104435197)

[5.1.1 Vstupné: 52](#_Toc104435198)

[5.1.2 Pomocné: 52](#_Toc104435199)

[5. 2.3 Výstupné: 52](#_Toc104435200)

[5.2 int pole\_org - definicia vstupných,pomocných a výstupných premenných 53](#_Toc104435201)

[5.2.1 Vstupné: 53](#_Toc104435202)

[5.2.2 Pomocné: 53](#_Toc104435203)

[5.2.3 Výstupné: 53](#_Toc104435204)

[5.3 void Radixsort - definicia vstupných,pomocných a výstupných premenných 54](#_Toc104435205)

[5.3.1 Vstupné: 54](#_Toc104435206)

[5.3.2 Pomocné: 54](#_Toc104435207)

[5.3.3 Výstupné: 54](#_Toc104435208)

[5.4 void Bubblesort - definicia vstupných,pomocných a výstupných premenných 55](#_Toc104435209)

[5.4.1 Vstupné: 55](#_Toc104435210)

[5.4. 2 Pomocné: 55](#_Toc104435211)

[5.4.3 Výstupné: 55](#_Toc104435212)

[5.5 void quickSort - definicia vstupných,pomocných a výstupných premenných 56](#_Toc104435213)

[5.5.1 Vstupné: 56](#_Toc104435214)

[5.5.2 Pomocné: 56](#_Toc104435215)

[5.5.3 Výstupné: 56](#_Toc104435216)

[5.6 int vyskyt - definicia vstupných,pomocných a výstupných premenných 57](#_Toc104435217)

[5.6.1 Vstupné: 57](#_Toc104435218)

[5.6.2 Pomocné: 57](#_Toc104435219)

[5.2.3 Výstupné: 57](#_Toc104435220)

[5.7 int vyskyt\_vypis - definicia vstupných,pomocných a výstupných premenných 58](#_Toc104435221)

[5.7.1 Vstupné: 58](#_Toc104435222)

[5.7.2 Pomocné: 58](#_Toc104435223)

[5.7:3 Výstupné: 58](#_Toc104435224)

[5.8 int switch\_volba - definicia vstupných,pomocných a výstupných premenných 59](#_Toc104435225)

[5.8.1 Výstupné: 59](#_Toc104435226)

[5.8.2 Pomocné: 59](#_Toc104435227)

[5.8.3 Výstupné: 59](#_Toc104435228)

[5.9 void export\_novy\_retazec - definicia vstupných,pomocných a výstupných premenných 60](#_Toc104435229)

[5.9.1 .Vstupné: 60](#_Toc104435230)

[5.9.2 Pomocné: 60](#_Toc104435231)

[5.10 void nahr\_interpunkcie - definicia vstupných,pomocných a výstupných premenných 61](#_Toc104435232)

[5.10.1 Vstupné: 61](#_Toc104435233)

[5.10.2 Pomocné: 61](#_Toc104435234)

[5.10.3 Výstupné: 61](#_Toc104435235)

[6. Podmienky vstupných, výstupných a pomocných premenných: 62](#_Toc104435236)

[6.1 void vymaz\_extra\_whitespaces - Podmienky vstupných, výstupných a pomocných premenných 62](#_Toc104435237)

[6.2 int pole\_org - Podmienky vstupných, výstupných a pomocných premenných 62](#_Toc104435238)

[6:3 void Radixsort - Podmienky vstupných, výstupných a pomocných premenných 63](#_Toc104435239)

[6.4 void Bubblesort - Podmienky vstupných, výstupných a pomocných premenných 64](#_Toc104435240)

[6.5 void quickSort - Podmienky vstupných, výstupných a pomocných premenných 65](#_Toc104435241)

[6.6 int vyskyt - Podmienky vstupných, výstupných a pomocných premenných 66](#_Toc104435242)

[6.7 int vyskyt\_vypis - Podmienky vstupných, výstupných a pomocných premenných 67](#_Toc104435243)

[6.8 int switch\_volba - Podmienky vstupných, výstupných a pomocných premenných 68](#_Toc104435244)

[6.9 void export\_novy\_retazec - Podmienky vstupných, výstupných a pomocných premenných 68](#_Toc104435245)

[6.10 void nahr\_interpunkcie - Podmienky vstupných, výstupných a pomocných premenných 69](#_Toc104435246)

[7. Vyhodnotenie časovej komplexity - Leto 70](#_Toc104435247)

[9.1 void vymaz\_extra\_whitespaces 70](#_Toc104435248)

[9.2 int pole\_org 70](#_Toc104435249)

[9.3 void Radixsort 70](#_Toc104435250)

[9.4 void Bubblesort 70](#_Toc104435251)

[9.5 void quickSort 71](#_Toc104435252)

[9.6 int vyskyt 71](#_Toc104435253)

[9.7 int vyskyt\_vypis 71](#_Toc104435254)

[9.8 int switch\_volba 71](#_Toc104435255)

[9.9 void export\_novy\_retazec 72](#_Toc104435256)

[9.10 void nahr\_interpunkcie 72](#_Toc104435257)

[8. Vyhodnotenie priestorovej komplexity -Leto 72](#_Toc104435258)

[10.1 Priestorová zložitosť funk. void vymaz\_extra\_whitespaces 72](#_Toc104435259)

[10.2 Priestorová zložitosť funk. int pole\_org 72](#_Toc104435260)

[10.3 Priestorová zložitosť funk. void Radixsort 73](#_Toc104435261)

[10.4 Priestorová zložitosť funk. void VymenStringy 73](#_Toc104435262)

[10.5 Priestorová zložitosť funk. void Bubblesort 73](#_Toc104435263)

[10.6 Priestorová zložitosť funk. void quickSort 73](#_Toc104435264)

[10.7 Priestorová zložitosť funk. int vyskyt 74](#_Toc104435265)

[10.8 Priestorová zložitosť funk. int vyskyt\_vypis 74](#_Toc104435266)

[10.9 Priestorová zložitosť funk. int switch\_volba 74](#_Toc104435267)

[10.10 Priestorová zložitosť funk. void export\_novy\_retazec 74](#_Toc104435268)

[10.11 Priestorová zložitosť funk. void nahr\_interpunkcie 74](#_Toc104435269)

[9. Vlastnosti. 75](#_Toc104435270)

[11.1 NUTNÉ VLASTNOSTI: 75](#_Toc104435271)

[11.2 ODPORÚČANÉ VLASTNOSTI: 76](#_Toc104435272)

[10. Záver 77](#_Toc104435273)

[11. Zdroje 78](#_Toc104435274)

# Letná analýza problému

V tejto kapitole rozoberáme efektívnosť jednotlivých funkcií problému abstraktne aj v detailoch v zdrojovom kóde, taktiež budú popísané rôzne verzie riešenia daného problému.

Táto analýza oproti tej zo zimy poskytne detailnejšie zameranie na časovú a priestorovú náročnosť a nebude sa venovať ošetrovaniu vstupov, vzhľadu a user-friendly prístupu.

## **1.1** Načítanie a vpisovanie do súboru a celkovo problematika vstupu a výstupu

Text

Description automatically generated

**V mojom programe sa nachádzajú 3 možnosti vstupu:**

1. Vstup priamo od použivateľa cez konzolu
2. Vstup zo súboru
3. Náhodná generácia s vopred zadefinovanou veľkostou TEST

### 1.1.1 Problematika alokácie

V mojom programe som zvolil statickú alokáciu.

Premenné sa prideľujú natrvalo. Pridelenie sa vykoná pred spustením programu.Táto alokácia zabranuje programu načítať viacej ako 20 000 slov pre využitie v programe. Limitácia je spôsobená implementáciou dvojrozmerného poľa ktoré sa alokuje pomocou násobenia riadkov a stĺpcov. V mojom prípade maximálna veľkosť jedného slova je definovaná na 49 znakov.

**Priemerná dľžka slov v angličtine je 4,9 znakov**, a najdlhšie slovo v angličtine je „Pneumonoultramicroscopicsilicovolcanoconiosis “ **a má 45 znakov**. Najdlhšie slovo v slovenčine je „najneskomercionalizovávateľnejšieho“ **a má 35 znakov. Na zákade týchto informácií som nastavil počet stĺpcov na 50 vrátane ukončovacieho charakteru \0.**

* **Pri prekročení dĺžky 49 znakov môžu nastať následné anomálie.**

**Podľa môjho testovania, abecedné usporiadanie fungovalo bez problémov, chyby sa zobrazili až pri výpise výskytu jednotlivých slov.**

A picture containing text

Description automatically generated

Tejto situácií sa zamozrejme chceme vyhnuť, preto statickú alokáciu odporúčam nepoužiť vo výslednom programe. Samozreme akákoľvek dynamická alokácia spôsobí menšie nároky na pamäť, ale bude to znamenať zvýšenie nárokov na processing a tým pádom sa aj časová náročnosť zvýši.

Statická alokácia pri náhodnej generácií znakov

Statická alokácia funguje tým istým spôsobom ako pri načítaní od použivateľa, len s tým rozdielom, že poznáme maximálnu dĺžku, ktorá je vopred stanovená makrom TEST, toto môžeme využiť na statickú alokáciu a vyhnutie sa zbytočnej akokácí miesta, ktoré nebude vôbec využité. Ale zvážiť musiíme aj ten fakt, že použivateľ programu nebude mať prístup k zdrojovému kódu, pre tento dôvod som od tejto verzie odstúpil, pretože testing slúži na vývoj a nie ako funkcionalita programu. Treba, ale taktiež dodať informáciu, že testovanie môže prekročiť maximálnu dĺžku 50 znakov pri dostatočne velkej šance nevygenerovania oddelovaciej medzeri. Tu môžu vzniknúť anomálie, treba si dať pozor a zadať dostatočný počet medzier do randomizovaného vstupu.

### 1.1.2 Dynamická alokácia

Hlavný problém pri implementácií dynamickej alokácií je chýbajúca infomácia o veľkosti .

Velkosť je známa iba pri náhonej generácií charakterov, čo ale nič nevyrieší. Po dlhom sedení nad Stack Overflow som ale našiel riešenie pomocou funkcionality realoc. Toto riešenie som implementoval, ale pri testovaní som dosiel na to, že to drasticky ovplyvnilo rýchlosť programu a pre tento dôvod som od dynamickej alokácií odstúpil. Je to problematika na další týždeň. A v momente ako toto zadanie píšem, tak už som 2 týždne po termíne. Vrátim sa k tomu ak bude čas a poskytnem dodatočné informácie k realizácií.

### 1.1.3 Výhody dynamickej alokácie:

* Znížené priestorové nároky.
* Možnosť pracovať zo vstupom vätčším ako 20 000 slov. Môj stack bol nastavený na maximálnu velkosť 20 000 slov, viacej slov spôsobilo crash, dynamická alokácia by odstránila tento problém.
* Uživateľ sa nemusí starať o vyhradenie priestoru.
* Uloží sa do pamäte iba to čo použivateľ potrebuje.
* Maximálna dlžka slov by bola odstránená

### 1.1.4 Nevýhody dynamickej alokácie:

* Zvýšená komplexita – časová
* Komplexnosť programu
* Zadanie by som odovzdal o mesiac

**1.1.5 Otázka pre profesorku, ktorú nemusí zodpovedať.**

Rád by som to konzultoval pri prezentácií zadania.

Využitie tejto funkcie v mojom programe... Bol by to dobrý nápad dynamicky alokovať pamäť týmto spôsobom? Ako vytvoriť optimálnu dynamickú alokáciu pre vstup, ktorého velkosť nepoznáme.

Potencionálny kód na dynamickú alokáciu:

**char** \*s\_new\_from\_stdin( **int** beExact, size\_t \*len, size\_t \*size )

{

**const** **int** AHEAD = 2; /\* # buffer pamäť? \*/

**int** i,j, c = EOF;

size\_t sz = 0;

**char** \*try = NULL;

**char** \*ret = NULL;

/\* Vytvorenie bufferu \*/

ret = malloc( AHEAD \* **sizeof**(**char**) );

**if** ( NULL == ret )

goto fail; /\* Stacilo by dať return 1 \*/

/\* čátaj vstup až kým nenájdeš koniec\*/

j = 1;

**for** (i=0; '\n' != (c=getchar()) && EOF != c; i++)

{

/\* Dalši buffer? \*/

**if** ( i == (j \* AHEAD)-1 ) /\* -1 je pre termináciu? '\0' \*/

{

try = realloc(ret, (++j) \* AHEAD);

**if** ( NULL == try )

{

free( ret );

goto fail; /\* Môžeme takto využívať goto? \*/

}

ret = try;

}

ret[i] = (**char**)c;

}

ret[i] = '\0'; /\* Ukončenie \*/

/\* ret je velkosť \*/

sz = (j \* AHEAD);

/\* ak je be Exact true, uvolni prebytočnú pamäť \*/

**if** ( beExact )

{

/\* realloc ret na presne 1 + strlen(ret) charov \*/

try = realloc( ret, (i+1) \* **sizeof**(**char**) );

**if** ( NULL == try )

goto fail;

ret = try;

/\* udate presnej velkosti ret na 1 + strlen(ret) \*/

sz = i+1;

}

/\* dĺžka a velkosť sa pošle uživatelovi\*/

**if** ( len ) \*len = i;

**if** ( size ) \*size = sz;

**return** ret;

fail:

**if** (len) \*len = 0;

**if** (size) \*size = 0;

**return** NULL;

### 1.1.6 Prečo som neimplementoval binárny súbor?

Pretože text zaberá rovnakú veľkosť v binárnej forme ako aj ASCII forme. Pre prácu s textom je implementácia binárneho súboru nepotrebná.

## 1.2 problém definícia „slov“ a ich selekcia

Tu slová oddelíme od nadbytočných medzier a po prípade, ak je zapnutý filter na interpunkciu, tak vymažeme aj tú a docielime tým správnu funkcionalitu algoritmu.

Čo definujeme ako slovo?

Slovo je podľa môjho algorimu akýkoľvek znak okrem whitespace, akonáhle máme medzeru tak definujeme to ako ukončovací znak.

Ako by sa táto funkcia dala optimalizovať? Podľa môjho hladania existuje iba tento spôsob, nemuselo by sa využiť dvojrozmerné pole, ktoré podľa informácií je na tom horšie z hladiska prístupu . Dala by sa tu využiť vstavaná funkcia strtok(), ktorá rozdelí string, ale podľa názoru ludí na StackOverflow je tá funkcia zbytočná. Rozhodol som sa ju nepoužiť, algoritmus tu zostáva rovnaký, len v tomto prípade je menej krát volaný a predstavuje zdokonalenú Radix verzia exportuje stringy v jednorozmernom poli na rozdiel od dvojrozmerného, čo v tejto funkcií znamená, že jej časová efektívnosť je horšia, ale stratený čas sa získa naspať pri sortovaní slov.

Táto funkcia funguje spôsobom , že akonáhle nájde whitespace slovo ukončí v stĺpci dvojrozmerného poľa vloží ukončovací charakter a pokračuje na ďalšie slovo = ďalší riadok pomocného pola.

## 1.3 problém abecedné zoradenie & sortovací algoritmus

Toto je časť programu kde som strávil 80% svojho času, môžeme totiž doladovať optimalizáciu vstupu výstupu, ale jadro programu pozostáva z toho, že použivateľ potrebuje abecedne zoradiť všetky slová v súbore. A na toto potrebujeme optimálny sortovací algoritmus, ktorý bude schopný pracovať s porovnávanním ASCII hodnôt.

### 1.3.1 Aký sortovací algoritmus som použil v zime?

Využil som Bubble sort, ktorý je jednoduchý na implementáciu, ale je veľmi náročný na processing a není vôbec optimálny.

Na malý počet prvkov ale stačí a preto som sa rozhodol tento algoritmus vylepšiť, tak aby bol volaný v programe iba raz a je implementovaný jednej z 3 verzií môjho programu.

Bubble sort ale nestačí na využie pri 100 000 znakov a viac, povedzme, že na prekonvertovanie knihy ktoré majú v priemere 90 000 slov = +- 441 000 znakov, tento algoritmus vôbec nestačil.

Časová náročnosť Bubble sortu : Najlepšia : O(n^2) Najhoršia : O(n^2) Priemerná : O(n^2)

Priestorová náročnosť: S(n)

Na záver môjho hladania som sa rozhodol, že využijem buď quicksort, ktorý je univerzálny, jeho princíp sa dá uplatniť pri triedení hocičoho a bude poskytovať skoro najlepšie výsledky, alebo využijem trošku menej známy algoritmus zvaný radix sort, v tomto prípade MSD radixsort.

V tejto časti by som rád sa poďakoval p. James Aspnes z Yale University v USA a Joshua Hug z Berkley University, ktorý poskytli úžasné materiály na výber a implementáciu správneho algoritmu.

### 1.3.2 QuickSort a Radix Sort v komplexite

**Quicksort**

Najhoršie možné scenário : O(n2) S(n)

Priemerné scenário : O(n\*log n)

Najlepšie možné scenário : O(n\*log n) S(log n)

**Radix Sort MSD**

**M** je dĺžka najdlhšieho stringu.

**B** je variácia vstupu, v tomto prípade ASCII má 256 hodnôt, B = 256

Najhoršie možné scenário : O(N\*M) S(N + MB)

Priemerné scenário : O(N+M)

Najlepšie možné scenário : O(N) S(N + MB)

Pre prípad, že nepoznáte radix sort, odporúčam si pozrieť toto video:

<https://www.youtube.com/watch?v=bg-qnrQS82I&t=1s>

### 1.3.3 Rozdelenie Radixsortu na LSD (Least Significant digit) A MSD (Most significant digit)

Najväčší rozdiel medzi LSD a MSD je v tom, že MSD považuje najvýznamnejšiu číslicu alebo znak ako prvú, čo prirodzene triedi reťazce bez iterovania cez všetky číslice v reťazcoch. To je výhoda. Rekurzívna implementácia MSD však využíva viac priestoru ako LSD.

Radix LSD aj MSD sú algoritmy triedenia reťazcov, založené skôr na takzvanom kľúčovom indexovanom počítaní než na porovnávaní. Najlepšia účinnosť porovnávania vždy bude O(n\*log n). Preto Radix dokáže byť výhodnejší a pri vyhovujúch podmienkach dokáže byť aj O(n).

MSD radix sort môže byť implementovaný ako stabilný algoritmus, ale vyžaduje použitie vyrovnávacej pamäte rovnakej veľkosti ako vstupné pole.

Táto dodatočná pamäť umožňuje skenovanie vstupnej vyrovnávacej pamäte od prvého prvku poľa po posledný a presúvanie prvkov poľa do cieľových zásobníkov v rovnakom poradí.

MSD Radix môže byť pomalší ako iné triediace algoritmy pri funkciách ako je napr zlučovacie triedenie a rýchle triedenie, ak sú operácie neefektívne. Tieto operácie zahŕňajú zoznamy podvložiek a funkcie odstraňovania a proces izolovania požadovaných číslic. Radixové triedenie je založené na číslach alebo písmenách a je menej flexibilné ako iné druhy. Ak sa zmení typ údajov, je radix nevyužiteľný.

### 1.3.4 Radixsort alebo Quicksort?

Quicksort šetrí miesto a dokončí sa len pomalšie o konštantnú hodnotu, Ak nastane situácia, že bude pomalší. Účinnosť zoradenia Radixu bude síce rýchlejšia ale priestorová náročnosť bude ohnoho viacej vytažená.

Radixsort je veľmi účinný ak zadané hodnoty sú typu int, v tomto prípade ASCII čísla sú od 0-256, čo znamená, že radix bude veľmi efektívny. MSD Sort začína od Začiatku, čo znamená, že vyhodnotenie trvá kratšie ako keď by algoritmus išiel od konca ako pri LSD.

Quicksort je algoritmus založený na porovnaní. To znamená, že jediná informácia, ktorú algoritmus používa o položkách, ktoré triedi, je návratová hodnota porovnávacej rutiny. To má pomerne peknú výhodu v tom, že je algoritmus veľmi všeobecný, ale má nevýhodu v tom, že algoritmus dokáže extrahovať iba jeden bit informácií z každého volania na porovnanie.Keďže existuje n! možných spôsobov, ako zmeniť poradie vstupnej sekvencie, to znamená, že potrebujeme aspoň log(n!) = O(n log n) volania na porovnanie, aby sme dokončili triedenie.

Ak triedime niečo ako reťazce, môže to byť obzvlášť drahé, pretože volania strcmp môžu trvať lineárne v dĺžke porovnávaných reťazcov. V najhoršom prípade triedenie n reťazcov s dĺžkou m môže trvať O(n m log n) času.

Ale cez všetky plusy radixsort je veľmi málo aplikovaný v reálnom svete hlavne kôli tomu, že nemá takú flexibititu a nedá sa aplikovať ako vstavaná funkcia, ale v mojom programe poskytne najlepšiu účinnosť.

### 1.3.5 Ako som sa výsledne rozhodol?

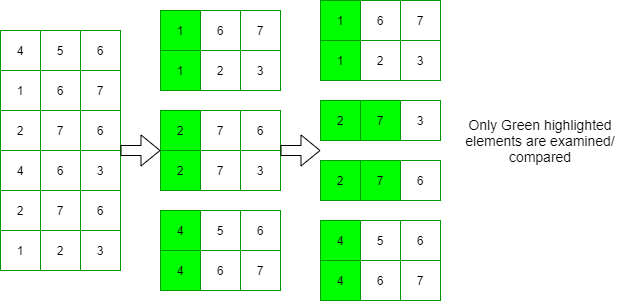
Výsledne som sa rozhodol implementovať radixsort ako môj výsledný algoritmus. Vyskúšal som taktiež aj quicksort ale dokumentácia k jeho implementácií spôsobila veľa stráveného času nad debugovaním bez konečného výsledku, málo ludí implementovalo quicksort na stringy a ascii hodnoty aj keď je najviac využívaný, vätčšina ludí používa vstavanú funkciu Q-sort, ktorá je vylepšená verzia quicksortu pre C. Nikto nechce znovu vyvíjať koleso.Radixsort je veľmi málo využívaný algoritmus na triedenie a práve to ma zaujalo. Veľmi sa mi páčili videá o jeho funkcionalite. A keď som našiel plnohodnotný guide, tak som nemohol odolať.

Videá o Radixsorte:

<https://www.youtube.com/watch?v=BeoCbJPuvSE>

<https://www.youtube.com/watch?v=kPRA0W1kECg>

Príklad funkčnosti:



### 1.3.6 Bubblesort LETO v mojom programe

**void** sorting(**int** space\_count\_prev,**char** dvroz[MAX\_SKEN\_ZAPIS+100][MAX\_SKEN\_ZAPIS+100])

{

**int** i,j = 0;

**char** tmp[MAX\_SKEN\_ZAPIS+10];

**for** (i = 0; i < space\_count\_prev; i++) // I SA MUSI = POCTU MEDZIER, POTOM SA SKONCI

{

**for** (j = i + 1; j <= space\_count\_prev; j++) // "j" MUSI BYT > POCTET MEDZIER, POTOM SA SKONCI

{

**if** ((strcmp(dvroz[i], dvroz[j]) > 0)) //AK SLOVO MA VATCSIU HODNOTU ASCII, TAK SI VYMENIA MIESTA, PRECHADZA SA VSETKYMI SLOVAMI

{

strcpy(tmp, dvroz[i]);

strcpy(dvroz[i], dvroz[j]);

strcpy(dvroz[j], tmp);

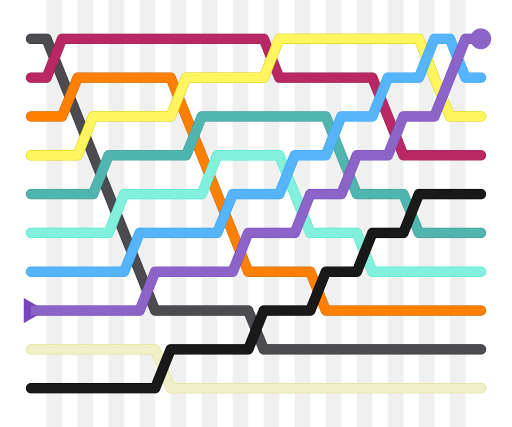
}

}

}

}

* Jednoduchý algoritmus bez dodatočnej komplexnosti.



Funguje spôsobom porovnávania, v prípade ak porovná 2 prvky a niesú v poradi, tak ich prehodí, postupuje takto dookola až kým neprejde celé pole.

Princíp je nasledovný :

Pre i od 1 po (počet prvkov)

Pre j od 1 po (počet prvkov - i)

Ak zoznam[j] > zoznam[j+1]

Vymeň zoznam[j] ↔ zoznam[j+1]

### 1.3.7 Quicksort v mojom programe

**void** quickSort(**char** dvroz[MAX\_SKEN\_ZAPIS][MAX\_SKEN], **int** lava, **int** prava)

{

**int** i, j;

**char** \*x;

**char** pom[MAX\_SKEN\_ZAPIS];

i = lava;

j = prava;

x = dvroz[(lava+prava)/2];

**do**

{

**while**((strcmp(dvroz[i],x) < 0) && (i < prava)) //POROVNANIE NAVRATOVEJ HODNOTY FUNKCIE STRCMP [i] PRVKU A [x] PRVKU

{

i++;

}

**while**((strcmp(dvroz[j],x) > 0) && (j > lava)) //POROVNANIE NAVRATOVEJ HODNOTY FUNKCIE STRCMP [i] PRVKU A [x] PRVKU

{

j--;

}

**if**(i <= j)

{

printf("\n\t prehod slova %d a %d",i,j);

strcpy(pom, dvroz[i]); //VÝMENA SLOV CEZ POM

strcpy(dvroz[i], dvroz[j]);

strcpy(dvroz[j], pom);

i++;

j--;

}

}

**while**(i <= j);

**if**(lava < j) //REKURZÍVNE VOLANIE

{

printf("\n\t VYKONA SAAA PRVE");

quickSort(dvroz, lava, j);

}

**if**(i < prava) //REKURZÍVNE VOLANIE

{

printf("\n\t VYKONA SAAA DRUHE");

quickSort(dvroz, i, prava);

}

}

QuickSort je algoritmus rozdeľovania a panovania. Vyberie prvok ako pivot a rozdelí dané pole okolo vybraného pivotu na 2 časti. Existuje mnoho rôznych verzií quickSortu, ktoré vyberajú pivot rôznymi spôsobmi. Môj vždy vyberá pivot v polovici. Kde je najoptimálnejší. Zistilo sa, že pre takúto verziu algorimu sa presne dá vypočítať vstupná hodnota pri ktorej sa quicksort chová O (n^2).Lepšia je randomizovaná verzia quicksortu.

### 1.3.8 Krokovanie Quicksortu.

Povedzme, že máme vstupný retazec: „**STROM PES MACKA ANDULKA**“

lava = 0;

prava = 0;

i = 0;

j = 3;

**x = 1; - STRED**

V prvej iterácií vstupujeme do **DO -WHILE cyklu**, kde sa porovná 0-tý prvok „STROM“ s 1-vým prvkom „PES“. Návratová hodnota je 1, ptretože PES je hodnotnejší v abecede. Cyklus sa vyhodnotí FALSE , ukončí sa a pokračujeme na 2 while cyklus.

2 while cyklus porovnáva návratovú hodnotu 3-tieho prvku „ANDULKA“ a 1-vého prvku PES.

ANDULKA je hodnotnejšia v abecede ako PES, cyklus sa vyhodnotí FALSE a ukončí sa.

V 3. WHILE Cykluse sa sa porovná hodnota 1<=2, čo sa vyhodnotí ako TRUE.

**Vymenia sa prvky STROM A ANDULKA.**

i sa nastaví na 1 -inkrementácia

j sa nastaví na 2 – dekrementácia

Opakuje sa vnútro **DO WHILE** cyklu.

1 WHILE cyklus – POROVNANIE PES A PES – FALSE - NAVRATOVA HODNOTA 0.

2 WHILE CYKLUS – POROVNANIE MACKA A PES – FALSE – NAVRATOVA HODNOTA 1.

**Vymenia sa prvky PES A MACKA.**

i sa nastaví na 2 -inkrementácia

j sa nastaví na 1 – dekrementácia

Akonáhle sa dostaneme za DO-WHILE cyklus, tak pokračujeme na **rekurzívne volania algoritmu.**

Text

Description automatically generated



V tejto časti porovnávame lavú stranu X = 0

* Tu sa výsledne j – dekrementuje. A prehodia sa slová na Pozícií 0 a 0 = ANDULKA A ANDULKA.



Následne porovnávame pravú stranu kde X = 2

* Tu sa výsledne j – dekrementuje. A prehodia sa slová na Pozícií 2 a 2 =MACKA A MACKA.



Následne sa algoritmus skončí.

### 1.3.9 Prečo quicksort dáva nesprávne honoty?

Túto otázku som ani ja nevyriešil, je dosť možné, že je to pre velký počet rekurzívnych volaní alebo logická chyba spôsobená nepozornostou, pretože jednotlivé pivoty si rozdelia prvky a zoradia ich abecedne, ale zostávajú tu neutriedené bloky slov.

Každopádne, rozhodol som sa neimplementovať quicksort ako hlavný algoritmus, je tu iba na porovnanie.

Dalej túto chybu v programe neriešim.

Ukážka chyby:

Text

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence



### 1.3.5 Radixsort v mojom programe

**void** radixSort(**int** n, **char** \*\*pole\_sort, **int** k)

{

printf("\n\n\t N je: %d \n\n",n);

**int** i;

**int** pocet[256]; /\* počet retazcov s daným znakom na pozícii k\*/

**int** zaznam; /\* zaznam najviac vyskytujuceho charakteru \*/

**char** \*\*kyblicek[256]; /\* bucket na znaky \*/

**char** \*\*vrchol[256];

**while**(n > 1)

{

printf("\n\n\t N je: %d \n\n",n);

/\* Pocitaj vyskyt kazdeho znaku \*/

memset(pocet, 0, **sizeof**(**int**)\*(256)); // Hodnota kazdeho elementu v poli pocet sa nastavi na 0

**for**(i = 0; i < n; i++)

{

pocet[(**unsigned** **char**) pole\_sort[i][k]]++;

}

/\* Najdi najviac vyskytujuci-sa nenulovy znak \*/

zaznam = 1;

**for**(i = 2; i < 256; i++)

{

**if**(pocet[i] > pocet[zaznam])

{

zaznam = i;

}

}

**if**(pocet[zaznam] < n)

{

/\* Vytvor kyblicek a vrcholove pozicie \*/

kyblicek[0] = vrchol[0] = pole\_sort;

**for**(i = 1; i < 256; i++)

{

vrchol[i] = kyblicek[i] = kyblicek[i-1] + pocet[i-1];

}

/\* zorad podla k-th znaku \*/

/\* Je to podobne ako algoritmus Holandskej vlaky - problem s farbami \*/

/\* Zacneme od spodu az kym nebude vsetko zoradene \*/

**for**(i = 0; i < 256; i++)

{

**while**(vrchol[i] < kyblicek[i] + pocet[i])

{

**if**((**unsigned** **char**) vrchol[i][0][k] == i)

{

/\* nechaj tak, pokracuj v kyblicku \*/

vrchol[i]++;

}

**else**

{

/\* Vymen s vrcholom vrchol vhodneho bloku \*/

VymenStringy(vrchol[i], vrchol[(**unsigned** **char**) vrchol[i][0][k]]++);

}

}

}

/\* Zoradili sme vsetko \*/

/\* Rekurzia na ovsetko okrem 0 a zaznam \*/

**for**(i = 1; i < 256; i++)

{

**if**(i != zaznam)

{

radixSort(pocet[i], kyblicek[i], k+1);

}

}

/\* koncova rekurzia na zaznam \*/

n = pocet[zaznam];

pole\_sort = kyblicek[zaznam];

k = k+1;

}

**else**

{

/\* koncova rekurzia na vsetko\*/

k = k+1;

}

}

}

### 3.3.6 Vysvetelnie Radixsortu.

Vysvetlím pri prezentácií zadania. Som veľmi pozadu s projektom a toto je tak na dalších 10 strán.

Dopíšem do budúceho týždna a zodpoviem otázky na skúške. :)

## 1.4 problém výskyt slov

Výskyt využíva správny počet medzier, ako aj organizované pole,zoradenie iba určuje postupnosť slov vo výskyte. Ak by sme funkcií na výskyt ako parameter dali vstupné pole, na funkčnosť by to rolu nehralo, jedine by to spôsobilo nealfabetické poradie ale skôr naopak: sekvenčné vo vzťahu s vstupom.

**int** vyskyt(**int** space\_count, **char** \*\*jednoroz,**char** vyskyt\_slov[MAX\_SKEN\_ZAPIS][MAX\_SKEN])

{

**int** pocet = 0, i, j=0,k=0;

**for** (i = 0; i <= space\_count; i++) // I MUSI BYT VATCSIE AKO POCET MEDZIER, POTOM SA SKONCI - KED SA SKONCI BUDU POROVNANE VSETKY SLOVA

{

**for** (j = 0; j <= space\_count; j++) // "j" MUSI BYT VATCSIE AKO POCET MEDZIER, POTOM SA SKONCI - KED SA SKONCI BUDE POROVNANE SLOVO NA INDEXE "i" ZO VSETKYMI SLOVAMI

{

**if** (i == j) //AK SA ROVNAJU PREKOPIRUJKU SA NA "i" DVOJROZMERNEHO POLA DVROZVYS

{

strcpy(vyskyt\_slov[k], jednoroz[i]);

k++;

pocet++; //POCET SA NAVYSI

**break**;

}

**else**

{

**if** (strcmp(vyskyt\_slov[j], jednoroz[i]) != 0) // AK SA NEROVNAJU TAK SA SLOVO ZAPISE A INDEX SA POSUNIE

**continue**;

**else**

**break**;

}

}

}

**return** pocet; //POCET SA EXPORTUJE NA POUZITIE

}

# 2. Opravené chyby v programe

* Opravená chyba s duplikáciou
* Opravená chyba s duplikovaným volaním funkcie vymaž extra whitespaces.
* Opravená chyba s duplikovaným volaním funkcie na zistenie počtu medzier (Táto funkcia bola úplne odstránená a táto funkcionalita sa pridala funkcii na organizáciu, ktorá podľa daľšieho skúmania túto funkcionalitu už obsahovala, stačilo pridať počítadlo).

 Sumarizácia môjho programu v zime:

## 2.1 CHYBA č. 1 - Duplikácia slov vo výpise

ZNÁZORNENIE:

Text

Description automatically generated

Chyba sa nachádzala v zle nastavenej funkcií porovnania 2 slov, krotá neregistrovala duplikácie.

Opravená verzia:

Text

Description automatically generated

ň



## 2.2 CHYBA č. 2 – VYMAZ WHITESPACES - DUPLIKOVANE VOLANIE

**Fukcia sa zavolala :**

10 krát pri zadaní jedného slova napr „test“ - 4 písmená

17 krát pri zadaní 2 rovnakých slov napr „test test“ - 9 písmen

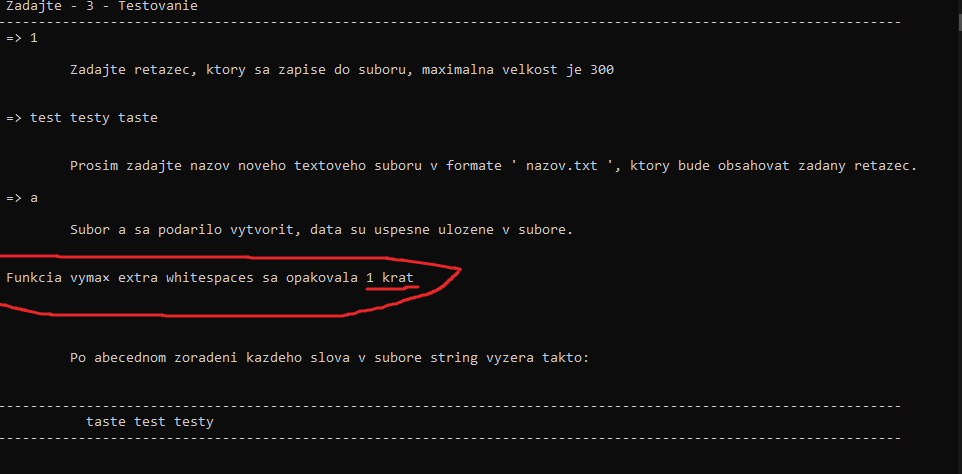
21 krát pri zadaní 2 rozdielnych slov napr „test testy“ - 10 písmen

25 krát pri zadaní 3 rovnakých slov napr „test test test“ - 14 písmen

31 krát pri zadaní 2 rovnakých slov a 1 rozdielneho napr „test test testy“ - 14 písmen

36 krát pri zadaní 3 rovnakých slov napr „test testy taste“ - 14 písmen

**Chyba bola odstránená, zbytočné volania boli zredukované na 1 volanie.**



....

## 2.3 CHYBA č. 3 – POCITADLO MEDZIER - DUPLIKOVANE VOLANIE

**Fukcia sa zavolala :**

8 krát pri zadaní jedného slova napr „test“ - 4 písmená

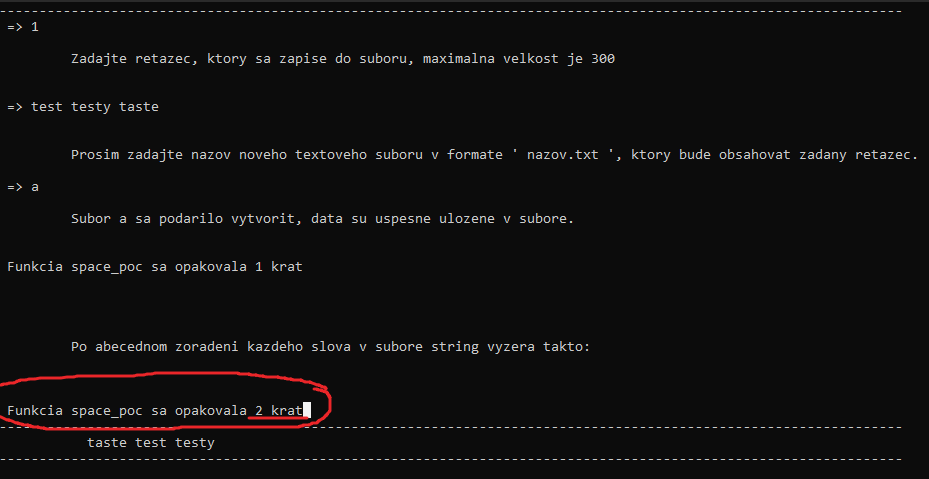
15 krát pri zadaní 2 rovnakých slov napr „test test“ - 9 písmen

19 krát pri zadaní 2 rozdielnych slov napr „test testy“ - 10 písmen

23 krát pri zadaní 3 rovnakých slov napr „test test test“ - 14 písmen

29 krát pri zadaní 2 rovnakých slov a 1 rozdielneho napr „test test testy“ - 14 písmen

**Chyba bola odstránená, zbytočné volania boli zredukované na 1 volanie.**



Tento problém som ale vyriešil nasledovným spôsobom, použil som funkciu na organizáciu slov, ktorá ráta počet slov vo vete a má už zabudovanú funkcionalitu toho, že nájde počet medzier, pretože medzi slovami je vždy iba 1 medzera, pri nájdení slova jednoducho nastavíme počítadlo medzier o 1 inkrement vyšie.

Bolo teda treba implementovať už iba počítadlo a potom následne zmeniť návratová hodnotu funkcie na „int“.

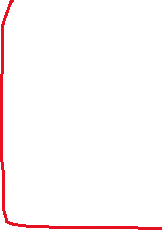
**Týmto spôsobom sme dosiahli získanie počtu medzer po sortingu.**

**Nová HYBRIDNÁ funkcia, ktorá ráta aj počet medzier, funkcia na získanie počtu medzier bola odstránená.**

**Pri volaní funkcie na organizáciu sa uloží hodnota do premennej.**

Text

Description automatically generated with medium confidence



## 2.4 Test volaní všetkých funkcií:

Text

Description automatically generated

+ Výskyt, ktorý sa volal vždy iba raz, pretože je viazaný na funkciu vypíš výskyt.

**PRIDANÁ FUNKCIONALITA v programe .**

* Pridaná funkcionalita výpisu slov s duplikáciou a bez duplikácie. – výpis.

A screenshot of a computer

Description automatically generated with medium confidence

* Výrazne lepší triediaci algoritmus, na výber je:

**Bubble sort -** **Worst Case** **O(n^2)** zo zimy, ktorý je jednoduchý ale v tomto prípade rýchlejší, ako bol v zime, hlavne kôli nadbytočným volaniam.

**Quick sort -** **Worst Case** **O(n^2)**, ktorý patrí k najefektívnejším, lepšie sú iba jeho zhybridnené verzie ako je napr TimSort v C++ (Tento algorimus není plne funkčný, opustilo sa od jeho finálnej implementácie, pretože algoritmus mi dáva nesprávne hodnoty. Skoro nikdy nedosahuje 0 - **n^2**.

**Most Significant Digit Radix Sort-** **Worst Case** **O(n\*m)**, **ktorý podla mojej analýzy skončil ako veľmi dobrý algorimus na sortovanie stringov aj, keď v realite není používaný pre jeho zvvýšené priestorové nároky a individuálnosť.**

# 3. MERANIA

Nasledové merania budú uskutočnené na testovaciom retazci a budú merať 3 funkcie:

Vymaž eštra whitespaces , pole organizácia a sorting.Nájdete ho aj v priečinku merania.

**Hardvér zariadenia, na ktorom bude algoritmus testovaný:**

**{**

**AMD Ryzen 2600 - 6 Jadro**

**32GB RAM DDR4**

**GTX 1080 8GB**

**}**

Testovací retazec 3:

**„**

, ac b, , a ,, ac a b,b c, bc a,,b ,a ,aa, b ,, bca b b,c c, ,ab , , c b a , , b b a , c b,, b acb a, b a ,b a b c b, bc,a,ac , aa, , ,, , , , ,c cc , c, a a , b , b , a b cc a b,b ab ca, b a a ,,, aca ,, b, , cbc bbc , , , bcbac aa b, c bb,b,,a ,ca , b , ccbb abacb c ac,, ac, c, ,a ,ac a bb , ,,b ,, , c ccca,, ,bc, b, , , aba bb c cbb,, c cb b, c, cc ,, b , b cc c , ac, c, aabc a ,, , c, b bbca ac ,,,,b, ac,,a,c c c a ,, b, ,,aa ,, ,cc aac , aa ,b b ,c ,, ,ca a a , a, c ,a c,aa ac ,c ,b b,b,cc ,,a ,c cb b c c, , , ,a a,c b b c , aaaa, a ,,a a a, , a, b ,c ,acb ,c cc b,, , ,,b cb c, , ab a , ,cbcb, b,, b a b,, ,bc, ,, cb,acb , ab cb ,c,,, a ,b bb , ,ca,,ccbb a, a bb ,bba,c, ,, , , , ,, c caac ,,a ,a ,aa ,bab ,, bb ,b ,,,ab b , , a ,,bac b , c b , a,a cccab, , bab aa c a,, a, , cc c, , ,bb ,c ba a,,c, b b ,b a, , ,b,c, c , c cb,, c , a,,, ,b b b,c b,,a , b ac c c, , ,,, a c b ,b b, a , a b b aab a, a ,,c, ab aac,c ,c , ,,,,b,, ,,, a , ca , b b b, , , ccbaa , b, bc b bc c , b , a, a ,a a , b ,c , aaa c bc , b b,,,c ,a ,, c, , , ,, , ,c, ccab,c, , ,, bc a , ,b c b, ,a a , a , c a ,c c,b,b , ba b bb ,b ba ,, a b b , ,ab ,acb ,,cb,a aab,, , ,a ,b,,, c a, b, a,b b , , c ,c c , ,,aa ,ca c a cbc, ,, ,c , ab , b ,, c ,,, a b ,c ,a a, ca c, ,c a , c b ,, a c bcca b , c b a c ,c bba,a,c , aaa cb, bca ,c c b ca , ,a c , ,c , , b, b ab , a b , , a , abca,, a a a , ,c,c, aa bc , ,,a a b ab , ,, ,,ca c bcc bb a, , ,b , a b c c,ba b c a ab, ,c, ,,b, c ,a, ,a b b c c,bb,, a abc,a , a ba , bb, ,b c ,a ,, c c c, ,a aa b, a , abc a , acc b, ca,c b b ,, c c ,b,,bc bc , b , ,a a,a c,,ca c ,, aa , c,c a, c, , b ,,, c, , b , , , b aa, ,b , c c c, ,, ,, ,c a, ,, a c b a,a a a c, , , ac b caa ,cca aac ca, , b a b ,caa,c a a cb, cb a,b a , , ab , a ,, ,cb,, c , ,, b,, , ,b ,cba,c ,ca ,,, a ab c , c, ,cc , b c b ab b , , a b, ,, aa c , bcb b , a b, b a cb b cc a , b , , , , ,,, b a c bc ab , bc c,,b,,a c b, , c,bbb a,,,,a,, c ,, , b ,ac, a ,,baa, , a ,c , , , c ,ca ,a c , a , c , , c c ,ba ,, , , , c,ca, ,a c ,, , ca a c ,,,, ,, bc a a , , b aab ,a c ,c ab a ba ,, , caabbbbb ,b,b b,c,b c a , ,b a,,,c c b c bbc, b a,,,c, c, a,,,c, ca c ab, a,abba,b ,, b ,,,,bba,bc ,,,, cc, a b cb , b, cc ca ca b , c ,b,c cca ,ab acc , ,a a c ,ba,b,a ,, , ,cc, a bb ,ca , a,ca,c a b b a,, a , a ,b b b c b, ,ba ,,b, c ,a , , ca, b b,cc b , , bc b, a,, c ,, ,c c,a b,a ba, bc ,ba,,a,,b, b abc , c a ,b c , c ac , ,,ba b, c,cc , ,cc ,, , b a a ,c , a,c cc, a,a aa c, b,ba , , cb,,ab, ,,b , c ,c,a , ,, , , ba, ,c ,ab a,cb ba ,bab,c ,, b b c cc, cc ,b,,, c ,b ,a, ,, ,, ac b , b abcc,,a,b a,b cc c c bcc b a,, c, b,c , ,,ab ,b,cb,b, b , bbcb bc , acc c a ,, b,c ,aa, c ,b a b b ac , a,,c c,a c aa ,c,, ,c c, c a b , , c , ,c a b cb,aa,a bba , ,cca a c , , ,b, b , aba,cb, , , cba , b b cc a, c,,c bc c , ,b acc , ccb bb ab a , , b c , c ab b,ba ,, b, ,ccac b b ,cb, a a,, a cb a ,a b a,c , a a c b bb cb, , b c, ,,b ab b b c, ,,b , b b, cc a, ,b ,a, a,cc b b, c , ,,b ,b c a c b c a ab,,a ,a,, c , , , b, b,b ca ac a b b ,, a c, ,,,,c,,, c, c a a , a , a ,,b , , ,c a ,bb ,a, , b a, b a , bbb , a c,c ac , a,ab ,, c,bc,,c ,b , ,,ca ,ba , ,, b,a, a,ba ,a , ,a ccc c , , c , a,c c, a , b, a a, ,,b b ,c ca,abb , a a b, a, b , ,c, , bca ,a b,baca a,, aa ,ba,,,, , bca a b ab , a a, c ba baa , b a,b c, b bc , , aca , acb, ,,b, c ,, caa b ac c , ,c ac,,cbbb , cc, c , , b , , ba,a,c bc c, b a, ,,b b , a bac c ,b,b c c ,b,, , , a, bc,,bb a ,,, , b ,, ,ab, c ,cb b c ,,, b , c ,cab , c cb,a,b a , , ,b c b , , c, ,b , c, c , ,,bc bcbab aa ,,,b ac cbc b ,, c bc , c, a b, cb ,abbc b, ,b, cac acbc c,, ,ac, , ,,b b,a b c, ,c b,,, , , cccaaa ba , a ,, cc,, , ,c b cc ca,, ,,c b ,c ,bb a c b, b , ,,ca, ,,,,b , aa,a, a ,,,babcc , , c , bb ,,, , ,aa c c,,c a cc a b a , , a, ,,aca,,aca ,a , bc cc,, , b ,c c b , caa ,c ab c a c , c b , c ,, cb cc,a a a c a,b,b , ,a, c cc,cc, , , c,cba ,b b,ccc, , ab,,c , ,, a c , , bc, ,ca a b,,,,b,c a ,c c ,c abb c ,c, c , c a , b c ,, acb , ,a c , ,, bc, , ,aa,c ,a,c a ,,ba , ,c b , , , a ,a,c ba c , , ,ccb ,, , , c,c c c,b,, , ,c, bc, c, bb,,ba aa,,,b b a,, bc,, a b ,a,,c c b c b a a ,aab, b, , a , b , ca b, b,aa ,c bbacbb a aa ,,b b ca ,,, , bcc,,, b, c b c,,,c aa, , , ,c bb, aa,cab c,,, , a ca , bc ,,,a b ab ,, ,b a, ,,, ,cc a,aa aba b , a a,cc b,, , cc ab a,bb ,ca c, , b ,a,, b, ,ab ,, ,, ,bc c , , ,c ,c, ,b ,, ,c , ,ac c,a c,, c c a aba aab , ,, a ca , , , ,b,,, ,cca ,a, , a ,c c,c ab, b, , a, , a bba,c, ,acbb , , , a ,cbb ,a a c , , , c c ,b a cb , acaba ba b ,c b ,c ,acc c,ba , ccc bab ,, ,,, c , aaa ,,cb b b, b c ,c aa , b ,a, , , , ,c,,, ,ca b ba, , bc a, b b, , , b b a a,, ,a a , a a ,ac ,b a a , , c c , ,, , ba, a a , , ab aab c , ,,,a c ca, c ,aab , ,b ,a ,,b c bab a ba c , a a ,, , ,, c , b,a ,,c , ,b a c ,ac ac, bbc ,, ca, bb a,b ,c ,b c , bb , ab,,, b ,, , bb,c ,b , ,bc c,b ,,ab,c, b , ,,,a, c cbb,b , ,cbaab, ,a b,, b ,a a, b ca,, ,b,b c c , bb ,b a acb,a ,c bc a b,,ab abca a a, b ,c c ,b , , cbacca b,aa b ab,ab b,b c, ab ,c,, cab , c, ,cb a c a,,ac ,, a , , a c a b ,a , ,, , , , , c b a ,b a , ba a abc ba b a, b , a ,b c b, c,, ,c , c , a a ,c ,a c , a cb , b aa b aa bc ac , c a, c b , ,bb ,c b, cb bc,,bb aa ,, ,c, , ,acbbac,cb ,b,ca , ca,a c b , ,cb,cc ac, c,, ,ab b b,,a ,,c ,, , ,a c aa,, c c , b , a ba b aa, c,, , a, , b, cb c a b,c c , a,ac,ba ac,,a ca a,c abb c ,b a,b c, , b, , , a , aabb, b ,,,c ,c,c a a c b,b,b ,,,,c ,,, aa, , a ,acc b a ,, , , a c ,, c , ab ,c ab ac, a, , , b , , baac b b ,b b ,a ,,ac b,, a b b, , a c , aa, ,b, a,b b b, a b , ,c c ,, a c ,, , ,a a , b ,b ,, a,babc a c,, bc b, a, c , a, a, b c b,,, aacc c,b ,a a b b a b ,,baaa, ,, a a b,,, , cc , ,,, c b , c , a,, ba ,bc , , ,baba, , ,c ac ,a ab b,c ,,a b c a c,b a,a cab c,,aa, cc a , ,,bccac b a , ab, , b ba ,ca c c,c, b , ,a,bbc,, b,, cac ac b aaab c a, b bcba a cbcb , ,cb , , , ,, cc , cc c , bbb c , ,a a c ,bac c b,cccc b b ca , cb , ab ,, bc b, c ,bcc a , b a , ,a,c ca , , b,, bac ,,cb,c a b, ba c bc, b ,c c,c b cc ,, ,, cc,b , bbca c , , ,,,cc b b b, ,,ab,c,, bc, aab , ba ccc,,accc , b,c b ,ba , ,a ,b ,a , bcb a a,b cc ,,bb a, ,c a ,c ,b a , ac ,a, cc ,, , b a,cb a ,c , ,,, b cca, aa cc ,a a a ,c , , c a aaa,, ,a ,baccb cb a b , ,,,,cb aa ,a a ab a,,a, b c ,ca,c b a ,a c , c cc,, a b ,c ,bb , aba,b b , ba , c,c, b,c,cc , bb,,,c ,,cb a a , , a,c b, c c , , ,, ac,b, ,a ,c b , ,a ,ca ,, ba c a b, c, , c, ac a b ,,,a ,a a a a a ab c b a b c ,,,acc b , , ac,c, a, a a a a, ,c, c , ,, c cbbc b , , b a b c ,c, a , b , , bb,ba bac a, , aaabc ,aa, a c, c ,b,a ,ac,ba bb a , ,b a c ,c ,c ,c,b , a a c aaa,b , ab, ab c bbca , ,, ac c a c ccbb , a ba,b ca, , c c , b ,,, b , , b bba,a,ac c , ,,b ,c ba , b , ac b ,bc,, , , , b ab, aa, ab, b , , c bc ,c, c aa,cc,a,,, ,ac ,c cb , b c, ,,, c, , , ,b c ,a ,c c, , ac ,b c a ,,, aa,a, ,,cca, ,,, ab c, , c, aba c b,baa a,, ,ca bb, ccc , ,abbc c,c a ccc ,a ,b , aa , , c, a , c ,,b c ,, c,cc ac,,, , , bcb ,, , b,a, ,, c a ab , ,c b , a a , c, a,c cb c a a, , a a b , , b bcaa,,,, ba c a, b,,c,ac,,,,, bb, b ,ba ,b , ca ,, , ,,c,, a, c,b, b b a ,c, bab, abc c, c a ac , ,c ccb,b , c, a,c , bb a cba,, c bc , caa b cb bb , ,,ab abc ,, , ,b,a , , aa,ac a c , b , , , ,, , , ,,c ac a bc c b,,, ,,, , b,b ,,, a , b a , a, ,,ab , ,c a a a c bcc , , b a c ac c c ,,ac c ,abb,, a b aa ,b,c,a c , b b ,b c, bc bc,b b, ,cb, ,aa ,a c b, ,b , , a a cbb ,c,,b ,, , b aa, b b a,,,, a ,a, bc ,ba c ,,b, b b, a a c,,a, ac ,,a b c , , , ,c , ,, a b , c,aa c caa , ba, ,, cc,a acb,,c bc ,, b,a ba a ,cb c aa c,,, b,,c, ,c c b b , a,a a ,bc , bb c b , ab ,a, c, b,, b b a ,, c acc , ,ba c ,c ba,b bc , c, , , ,c c a c bc ,, b a,aa , ac b ,, ,,abb ,,c , ab,a ,a aac, cbc ,c bb, aa cb ca , a ,ac ,bc ac aba c b,a , cb,, , ,cb ,b , a ,, a c c , ,, , c, ,a,bb bb , ,, ,, c bc ,,,,c ,a bca , bc ccc ca , baa cc, ccba cb, ac,,, cb , b,,aacc,b , ,,c, cc, , ,b cc aa a, ,ca , a ,cc , ca, , bb b , c, ca a, , bc b , ca ,bb,,,ba a bb,bab abca, , bba aca cc ,b ba ab,c , c,bcc, b a , c a c , a , b, ,b a b,bc ,cc b ,,acb,ba , ,, c,a,, b,,b,acb , a,, ,,b ,,cb ab abac a a, c bc , c a , b cb ,b b ,bbb,,,, b a ,,c, , b ,,a, b b a, , ,, c b , ,, cc b ,, a , ,, , a b, , ba b c ,,cc,,b ,bc ccc c,c,b ,ba bb , b ,a a ca ,,,, ,b,b c c cb ,a,,c c , c a ,b ,, c ab b b, , bc , ,cb cb b b bcb, , ba ,b a ba , a,, c b ,c , c,, cb, , a a,b,b aa,,c,,a ac b, b a , c,c, ,,c b c,cab bbbc , a, a,,c b ac , a, , ,c c , c b bc bcc , c ,, , c ,,,bb a c caac a a ,bc c b b c ca a a ,, a, , ,a , ac c a,cc, , a , a b bb , ,bc ,,, ac bab a bb b,ab, b a,aacc,b ,a, c, b ,b c ,,, bb b ,b, c ,c a cc, c , c ,, ,,a bc ,b c b b b c ,, c ab ,, ,, ab cb b b,c , ba cbcbcca , , a,, ,a,a, , ,,c a b b ac b,c , , b, c ,a b,, bb c ,, c, , ,a,b b a, b,a a ,, c , ,,, ,ab, ,,,,,bba,b,baa cc,,a ,b, b , ba ac, , a , c c, ,caa a,b a , , a a cccb , ,b aa ca acb a , b,b a c b ,,c c , ,b,,c c,,ccc,ccba, a,a, ba b a a a ,a c b,,a ,, , , ,b cc c , a , c c b b cca,b a ,,c c,,b , c ,b a c, , acb ,a a b bb ,bb,a, ,a , ,c , ,a bc, , , ca , ,cc a ,c c ac ba , a a , a, c , c cb bb , a, ca, b c, , b cc , b a,a cc a bb a a , ,bc, , a a , c bb, c, ,a a, cc ,, ac, c c a,a,c c , , , cb b, ,, , b b cb , a ,bb ,b ,, c c b b ,,, ,, b a, aa b,a, ,,a ,, c ,, c ,,a,bbc c, ,b c, a,cb, b a, , , , , ,b bcbb a , , b a,c,a,,cca a , ba, ,ab,c b ,,c a abc, , ,a a,aabbb , , , c ccc, , c, c,a bb bb,aaa,a a,a ,b , c a ,,b aac b bbcb b bbc b ac ,a a , , , a c c ,,,b b,aac, ca a cc c ,,a , ,ac,b, ,, ,b b,b a,, a a ,b,ab,a,b c ,,cb c ,ca, , ,bca a ,, c ba,, c , ,aa, ,aa,,,ab,,,,, cac cc a,,,a, c a ,,, ,, , c aa b , bb ,cb b c,, b b , a , ,a ,b ba b b, b, ,a, , ac a ccab , c c cca b a a , a a , a abb aa , ba, , a, , , ,b b cc , , a,b, b ,,c ,b c ,, ,,b, , aacc a , , , ,b c , ,, ,, b ,c, , cb, , cba c , bc, ,b a, ca a b cac c aac , a , , ,b bbb a b ab b c a, , ,,,caa ,b c, ,aa bc ccbbb ca , a c , ,, a a b cc c ,, c , cb , bc b ,b , , ,,,, c a c c a, b, ,a b ,,b ,ab , a, c a, c,, a a bc b aa , cb,, a c ,,a b c a bcbc , a c b , cc b c bbca ,b,,cc,,,aaabbc,bbca,,, ,bc , c,b b ,ab b b a c b aa ,ca bb c bb,, a ,bc,, ,, ,accc,,bb ac ca a, a ,aca ,a ,, c a b, b b ,,, , b,bba ab, ca c, ,c , c b bb cac ,, , ,, c c , ,, a,,,c, c ,a, b b , b b,ac, a , ba , ,,b b,ba, ab a b ,a,,ac c, , , ,,aba cbc ,,a , bc ,,a c cc,, ,,b abb, a , , b b c ,ca, ,,a , c a,cb, , , , , ac c ,b a , , ,,,, ,,b , bc ,bc ccb , a c b, ,b,,b ,b ,,,a c , ac b , , a cb cb , ,, , ac , ,c,ab ,b c,a ,, , , bb , c ,, b , c bcaa c ,c a c b,,,, a,abb , , c bbb a , b c c, c,ba ,, , b a c ,cac,,cbca ,, c a ,, , b a, ,, ,c c, ba c b , c ,a,,ab c, ,,, ,ca c,, bac b b,a aacbab, c c bb c ,, ba ca,aac bc bc ,bb c ,,cba, b , c c , a, b b , c, a cc , ,, , , b,b ,c ,b b, a a b c cb aa,a, aaa , bb,c,a caac ,a b,b, a, c a b,,, , ,a , ,, , bac ac , acc , b a, , a ,, ,c b ,cbbb, a,c ab, b, cba, b a , , , a ab ab , , b c a cbc ,, bc , ,, cb a ,a , ,c, c b a , b bca ,c, , b,b ,,c, , b b b b ,,,aa c a b, ba b,,ca, , c,aa, ,, , b ,bbab ca ,a c ab a b a a bb c,aacb,b a,,a ,,a , c , , , ,aa c cb b, abab ,b, , b b b ca , c bc ,, b b,b ab , ,, a,c b c ,ca, ,,c, a,b, ac a , b c b a , , , c b ,b c ,, , , c , ,b b c, ac b, b ,a,c, , c,c , c , ,, ,a,,a c,bc a b , a b, , ,a,, a, ,cba, , , ,,,b, ,b,,, bbc ,b ccaa bb a c a a c , a, a a,bc c ab, c ,,,b , a c,aa , a a b, ,,, ba bc, ,a a , ,b,cc , aac, ,b , a a, bc , , , ,a c,,, c,acb a,c a ,b cac c,, c,b ba a,, b, a , b , , ,a a c c ,a cc baa ,a bba , , ba, , , ,,,, ,c a ,a ,b a b b ac cb c,, , a , ,ac b bca b , , a ,c ca , ,ab c,,, , , a , , ,ba b ,a, ac, ab,a,cc a c a c c , ba ab bc, cc , ,a,, ab ,cba ,b,b a,b , c ,,b,cbc , bb a ,, a c ,, b ,,,a , b c, cab aca b, c , a, a,c b,, c ,,b c,ba b ,aab ba,b c b c cbb c , b a, ac , cacc,ac acb , ,, ,,a , , , , , , ,,, c,b ,ab , c ab ,b, ,a b , , ,ac, ,, b cca,c a ccb, ,,c ,,cb b ,,b c ,c ,,a b a b ac bc ,a abab , , ,, a , a,,,bb ccc, , b, bcc,a ac b ,, bc, ,c ,, , c ac,,, ,, b c ,, , cbb a c ,c, , c b bc b c , cac , b a c b,,a cc , c,b b ,b , ,a a c abbb c , , b,b b b , , , , a,b ab,a ,b , c ,,aacac,,b,a aab ,ac b, ba, ,c bcbb,b, , ac b c , bc , , ab,, , c , , a ,,,b ,, aac,cc,b cc ,bb , ,, c,, a bbb, ,cb, bac,, ,, aa ac c c,,ca b,b a a ,, , ac c ,, b , ca,a b,a ,,, ,b,, ,,, , ,, , ,c ,c,c bc c baa c ,caa,, cb b,b a a cb aaa a b b, , cc b,, ,c, a, b a cca bb ,bcab b c c b bb, ,,, ba,, , ,,a bc acca c b,baa ,c, , ,b bc ,ca , b , a,cca bbb,, b a,a , , c , bb , , , ccbac c, a,ba b,bc , ,, a, b , ,b,cc c ,b a a a b ,a ,bc c ac ,,,ba , c cb,, , bb,, ,a cbcbaa,b bc b, , a b,baa,ba , c,,ab , cb,,c c ,c,,c ,, a, ,,aca,c, c c b c c b ,b b,,,,b ca b, a, a,ba ,a , abb abbac,, cb, , ca, c b , ba ,, , a a c c c cb b, ,b , b b, b bc bccb b ,c,ccb , a c c, a c ,bb a a a,b cb a, c b , b b, aab ,,,,, ,,, ab c baaac ,b ,b c,b a, , c, ,b a c ba,b , b bc , , a ,a c b , ,cc b,ac b a,, , , c c b, cab c a ,b,a a b c, c c c c a,, , a,,, b cc cac ,cb , ,,, b,b,, bc b b a c cb ,, ,c c b ,, acc , bba , b ccac c ,, b c c ,,c aa ba a , b , b, ,, c bb,,ba ba , , ,,c,a, a ,cb, a , ,,b ,b, a b,,,b b ab a,c , b, a , c,a a c cb cccc ,,c , ,ab bc ,b, a a, b, , ,,b b b b c bcc, c ,, aaa a ac b ,b,c , c , b b , ,ba, c,b, a c ,a cc, c c , , ca, a c c ca,b ,aca , c , aabac ba b ,, ,ca b a a a b, ,b ,,,, c, a ba , aa , , b , b,bbaa c,, c c , ,bb b,,, c ,a ,c ,a , ,ba a,a a , c ,b , ,, a,a, c c c aba bba, , ,,a b, ,c , ,b c ccb ,bb ,b, b c a , ca b , a c ,, c b, ,,c ,c, , ,,b b a , a a bac c b ,, ,, c, ,,a, c,c, , , ,, , bc,a a b a ab ,c,,cb b , b,a b,,a , , , a ,b a b, ,,ba bbc ,,ba b,ab b b c ,ca,, , abb bc c,c ,c a b , a b , b, ,, , b b ,,a ab, c, a,, , ,bca , b, , b b, cac , b , abba a a b aa, cb b c b,,, a cab , bc cb,a c b ,, b, a b ,c , c ,cbab ,b , ,a, , c , b , b,bc,b ,c aa c caa,ac,, a, acc ,ca a , ,,,a,a cb bb c , c b b b , c ,b c c,aa , , , a aab c bb ,,a , ,b , b, ca ,a, , b, , cc ab c b , a c, b, cc c ,,, , , , ,b b a ,, aab c b b c bb b b, ab,a c ,b, , a a, a a , b a, c a ,a b c c ,, ca , c,b , bc a ba c b a c aa,,c ca ,cc, c , , ,b , ,ac ba b ,c a, a , c b,c c, a ,,, b , ,b a b a b , b,,a, c ,,a a, ,ca, b b a ,b, a , bb,c a cc acb c , a bc b, , bbcc b , ,b ccc cbcb , , cbb c, ,b,bc aaa abcb ,cc , , , b a, , , a , cc, ,ab aa , cc c b b,,, b b ,acc,c cb ,,b, a bab ca,,cab b c a a, ,aa , a , ,bc,,, c, a cac ,, a , , , c ,c , c c , , c , b, c, , ,b b,c a b, a,,a ac ,, b, c c ,aabb , a,b, b acb ,b , a b ,, , c , bb ab acb,, a ,c, ,c a ,ba a,,ca, b , , a,ccb,bbc, a,ab, ,,a , bc a ca a c,c b c a, ,b,b ac , ba, ,, a a, c , a,cca baa ,ab , , b ba b a , b c, b c b c,a c , , ,,a a bbc , a bb, c,,, , c , ,a c , abc , abca b ,ab , ,bbac, cb b , ,, c, , ,b, ,ccc,c , c b ,,, ac, ba a,b c, , b ,c a ,,bb c , c a,,abb bb, , , aabb, ,cbc b c ab, ,, ,ac c ,b b, , ac b ,c a , , ,a , ac , ,, a a, c ab ,c , ,c, c cac , a cc ab a b a,,, ,a a bccc , b , aa bb a acca b b ,, a,, c,a b a ,cb,a,a ,, a b b a c,,a ca b, ,, c c acc a, , a ,bcc , a b c , , b,a a , a, b b,c b b a, ba cb,c bb acb,,ca ,b, a, , ba bc,bc ,, c , , c , c a,ca, b c cc c c ca ba b aa , ,cab,c, b bba,a ac ,, ,,,bbab c a, ccb c,bc a c , , , b c , b c ca c c, a a b,,b, ca , b, , , , bbc cc ,b, a, ,,a ,a ,,c , , c, cca , ,, ,bc ccba, b ab c, , aca,aab c c,b a, , bb , ,c b, ,bb cc , a , , , bbaca, , cc , c ,,a cba ,c , b b,c, b,bbcbbc ba a, ,, ,a c, ,c bccc c a a , c ,,c b, , , bc, , ,aaa, ,, ,aa,, , aa bc ,acba c b c,a b , , b a a , , a a, ,, a abb a a,, ,cccb , bac ab,, ab b ,,b abb ,a b ,cca b b,a ,, , , ,b , , a, , a,b c c , a , c a baa c, abb a, c ba,, b, ,a, a a,, ,bb a,cb,, b a a, ,b,c c ccc ,cb a bb, ba , ,,,a ab,c,a a ,b , , c a , ,c cb a b,,, ac cc aa,,a a bc ,,,,a , c acb , c c, c b b, , a , cccca,, c c , b a, b, , c c, a , , a,b , ,a ,, ,b ,c,ab a ac,, acb a cb b bca bc ba bbb , a , c,, ,c ,,, , c ,b c ,b, , ac , c ,c c ,b a, a , baba,, a , b,c c ,, c , ,c ,cac ,,b a ,a aa, b ,,,ab ,, ,c b , , ,, , , b bc c cb bbb, cac cabc b b , , ac, , cca ,c , b b, b a c, ,c ,c c b c, cc b,,c , , a , b bc a , ,, c,bb, b , a, , , a bc ca a b aca ca , ,,bca c,, ,bb b ,,, c,, c,,b , c a c c,c, b bc c b b , ,ca, ccbb, ac b ,b b , , a , ,,a , b ac,a a aa, bb b ab c a b aa ,, a,b b c,caacc a c b, , b , c c c baba,ca , b c ab cb, bb, c , , , , , , , a, ,, ,cb , bc, cc b, cb b c cc ,aa ,,,, a,,ba,b cbcac aaa ac,c , a , , , b a ,ac,aab, , ,b cab b ,,,,, c cc b, ,b ,b, a a a baaa c,,,,a c, a baa ac,b a, ,ca c, a ccc b,,b , a,b,ba, cc,b , b bc bc c, ccb, , b,ab b ,,c a, aa, ab , a a a, cbbc aaac c, cc a c a, bc ac a,,c a , b , cca , c ba cc, , a , b , a ab,a c c,, , cb , , , ccaa c a ,ab ,b , c ,,bb a c bc , , ab a a , , bab a , a, , , , b,, cb b b ba,a, a, bba b , aa , , b, b ,c c ,b, bc c, cbc , ac a , ,ba ,,b , , a , , , , b ,c, b ,, , a a,b b c a b c baab b ,,b ,c ,, ,, c b b, , a b , , bb,, ,c,, cba b, , , a a,a c ,ca a ba, c , b,, ,,b cb c c a, , a, a , caa, ba,b,c b, a bba , , c c,a c,,bb, b aa ,caa c acbb ,,,, b c a , ,, ,aba a b ,, ccb,,c , ,,,aa bca , b aa , a,, bb , a ,b, b c , b b ,, , c c, ,a ,a , , ,,, c ,b , ,a ,b, c cb ,,ab , , , ccb, , a ,, , , a c,b, ac, a , c ,ab,, b,,a,b , a a ccabc ,cc , b c b,ca, b ,c, cb , a, ,,a ,,a , c ,ac abaa c a a ,,a, , , b a,,ab,c caa,c ,, ,,,aa a , c , bac , ,aa , b , aa b, c , ,, ,a aacc, ,,,a b ac ,,, bb, c,a a, bcac b a,a baa ,c b b cb c,, a b,,, ,a,a b,a ,,c, ,c c a a c, b,aa , , , b , c, , b b ,, b , b , b,b bab , b, b,,,b a , a,, c b c ,c b , ,ccb b c , , ab,b , ,aca , a c b ,, b,b b,,, a ,a,c a c ,a,c, ab,, , c c,,,a a , ,accb , cbb a aba , c,caa b,c,c ab a a c ,c a c , ,c b a , c ,,c ,a ,b , b, ,aaa,b bc , aa b ,ac ,b b ,, ,, aab c a c,,a, ,,,ac a c b,,b ,a, bb c,c,b b , cc ,b, b, b,, , , ab a b ,a , c bb,a , b,,,b a cb , ,a cb abc b aa b , c c,, , ,bc b,,b, ,b, , a a ,, c ,a ,b c b a c , c b,cb , b, , , , acb a b ccab, cab,ab c ,,bb ,b ,c c b,a , ,,, cacbbc , , , a , ab, ,, c, bb,,caa, , c , acb , ,, ,, a,a, ,,bac c , ,,,,c , cac c,, , b , a ,ba, , bc,,a b, c c,,, , c a aa ,c , ,cca, ,aba,c, ,c c ab , , b , b ,b , b ,,a,ac b,a c b,b,,,,cb a,c, c , a ac bc , c ,b,, c,,, a,c b c b cc, , ,,,c,b , a a, ,bc b ba ,, b, ca , c aba ,,ccc , ,ac , , b a ,abaa, a, , ,,, a,cc,abcc , c , a , cca ,c b b bc , b ,c , , bcc , c, b a ,ba b bc b a c c c c c a c ,, ,bccb , b,, b , a ,, a,,a b , c b, c,c,c , ,cb bacc c,a cb ,, , ,, ,, , , , a b, c c b , b ,c ac b , ,,a , a, b ab, ,b , ba b,,c c b ,,cc , c c,,,,a ,, ,b ,,, baa a , b,, ,a a bb c , c, bcbbc, bc ,a,ba b c ca, b,b , , ,c ,,ab bcb bcbb c , a, a c , , ca,a bc , , aa b bc, , , c , a, b cac, , ,c ,c ,,b ,a , ba a, b , b ,, b, a b b,,bc, a bc c a cc,,, a b ,,c c, b,b ,cba bc,a , , , b c,a,,,b bc , , a, c ,,,b a cbb ,,,a cb ,,,b b , b aa b ac a ac a , a a,,,aac baccb c, ,bb ,bbbb , ,b bcacc ba,a aa, b b b, c, bc,c b, cb ,a ccb , b c, b ,ab a , bb, b , ,c , , b, ,, aa, b , , c a cac,c, a, b acb ca , b cb ab ,a c c c c b,b, , c ,baa,, ba , ,,a, , , , b, c b ,, aa b c b, bb , bc, ba , , c, c a ,a b, a,, aa a bb , ,a aa b bb b,, , ,,a a,b ,,, ,,, ,b ,c ,, b b a , ,, a, a b b bc bc b,,ac, a c ba abba c, ,ca ,ba,bcb,,c ,,, , b a, b,a a, cac,c ,c ,,, , c c , c a ,caa c b b,, a b c, c a, b , , ,c,,,b cb, c a a c,ac c , bac, b , a,, c a a,c ca ,cc, bca, c ccb ca ,bc, b ac ba c,b cbb c b cb c c ,c, bb, a ,, a b ba a, c,,bbbba a ,c,b ,,,b , , ,,c b, c ,, ,,a, , , ac ,bc bc , b aa c , ,b ,, ccab ,,,c , ,b , , ,,a, c b , b, cb a c c , c a,b,ab , b, b,, a c b, b,ca ,c,c,,,ba b,a b,,,, , c,,a ,, ,c, b c , c,a, , cc b, b ,cccc,b,, b b , c b a ,cb b ,, c b b,b , b , a,,bba b, ,,cb b,b ,bc, babcc, b,b a,, , bc c,c b c ccc bc c,, c ,,,,, ,cc ,, a a,, ca ca cc c caa c , c,b b bcaa c ac ,c , , ,acc acab, , b, a, a b ,b a, cbb aa ,b b, b, ,b ,a , , ,bba , ,a , b, ,, a a a, a b ,a cb, , ,,bc ,,b , c ,a, c, , b , , , ab ca cb cb, ,, b, ,c bcc, c , ,, ccaac,b, ,,,c c b, b,,bcc,c b , ac c c b , b c,,,,cca,c, ,a a,, a , a, c,b a, , ,, bcbbc a a b c ,, , b bb, ,, ,, ba, , ,, aab bb,a b, ,a b c a ,b a c b c, , ,,c cb a a, , c a,c , ,, ,, ab bc,c abc ,,b ba,b , b, c c ,,ab a b, bac, , , bc,bcccca ,b bb , ccc b caa a ,b, a b aa c a , b c a b ,, a ,aa , ,c c ,,,ac, a ca a b ,, ac,c, , , c b a, a c bc, ,,, c,c, c c,, ac a ca , b,c c b, c c , c ,c ,a,bba a bb c,a, c b ac, b , a ,b ac a c , bca a c bb,c , ba , c, b,a , ab,a c a a ,, , a, , ba,, ,, cb, ,c ,c ,bbb bcb a a b,,bbcb c ca , ,b ,,cccbc b a ,, , ,b ba , c b , b b a , c, a ab c a ,bc , bbb ,b a , , cc bb, c c, , c, c c , cb c c a c, aba b c , ,, ac,,b b c b a ,c ,a cc,,c,,, ,c ac,, ,, a c ,aac a,a b , , a, b,, a, , a,cbb ,,, , c, , ,cc bc a , , ,, a aa a,ccccb ,, ,ca ,, , a , , ,, , ca, abb cacbcc a b a,, c ,b , , c a ,c ,, , a,b ,, ba c, , c,, cac,a , ,,c a ,c,,a, , , , b ,a, b b ab b c ,, b, , ba c a ,ba c a, acb a c c a , abb , , ,ca ,b a a ,,b c ,c c cb ,, c a , cb , , ,,,c c ,b c ,c aa ,a, bbc accc, b , c b, ,c ,a b, ,b ab ca,a , , cb c aa , , ,cbc ba b ,a ,b b a, ,aa c , ccb a ab, ac, c c,, bc , ,, ,b, ,,,bc,,b ,a b ,b , c b ac , ,c c ,,a, c ,, ba, a ba c , ,,a c ac, b, aa, ab, c, ,aa,, bb a a,ba b a c ,, , a b,c ba a bbaa a c,bb b c b a, b , a b , ,b , cba c bcc b, ,a, ,ba c ,c b a ccabb a, cb , c ,aa b , , ba b,ca b , c b c a a c,, , a c c b, , a,c , c b c , , , c , b,a a, ,bbb ,a, ,cc, ,, c bbc baa a ,, ,,b , c ,a c,, b a,, ac , c aa b , , a ,a , , c caa, b a,a a , ,c a a, b a bac, bb ,, ,,c cca b ,,,b c c,, c , ab b ,b,, ,b ,bbcaa ,a cb, ,ba, b c,, cb , ,c,, ,a b a ,,,b ,, cc,, ,,a , b b ,, c,,,, c , bc, ac,, aa b , ,, ,bb, ,, , a, ,b,c,, , aac c ,bb , ,b b ,bb, ,, ,, ,, c , , aa a b ac b,, c a, b,c c abca, ,,b bb b , b, a a b ,, ,a , ba , ,, ,, , a a, aa c ,c c ab, b b ,,b, aa ,, bcbb,a, a , bc,b ,,b c , a,c c ,, , bca,, bc , b , a a ,ab,, ,c , a,,,a , ,bc ,c abc ab,bb b b c ,b, ,b c aa , , , cc, b c , bc ,c ,,,b a ,b, a cac , ac ,bcb , a b c, a,a a a , b b c b ,, ca,, c bb b ,aa ,,,a, abcaaab, , , a c, , , bbbab a cb bc ,a ca cc, b ccb cc, b , b, a ,b,cc b,a ,,, c , b , ,a,, , a b ,,b ,, c , c,abb, b a, , ,, a c , ,c ,,, , ,a , c, c ba c, ,ac ,b bb c, , , ,b , b a ,,,,,a, ,, , ,aa caa c b b a a,,a,a , ,, , ccb, aa,b , b ca cccc a , ,b ,a ,,a a c ,b ac a c, c, b bbab b cb ,b,a ,ab b, ,,,,cb, cb b,, c,, b ,, ab ba, ,bb ba c, c b,b cb, a c, a c, , a ,,cbb a , b c , , a,aa ba c, ,c, ac ,aca ,, c c , bca, , bccb,b ,b ,c , a b a b ,c,a,a c c , ,bb c a c,b,c a ,c ,ac, c,, ac ,, c ab ,, ,c a ,, cc b ,b , cb bccb, ,,a,c , ac bb b c ,b ab ba ba,, ,c b, c a a ,a b ac a,bb, , a ba c bb , a b b ,aacc, , c a , c , a, ba , abab ,a ,a, a bb c a a b , ,b ,a ,ab,,ac , , c ,,c, a , bb c c, a b, ,bc , aaca a c,c c ,,a,b , ac a , c, b,,cb ,,b a b, bb,cc c a, , c ,c,, ,ab,aa aabaa a , , ca , c c b, ,,c , c ,, b ,b c,b b,,,, a cc ,, c ,, ca , a ,c, ,,b,b b a ,c c, c b c c ,,a ,c b c , b ,b , bb,,, ba, c b c, bc ,,, ,b, c b cb, ,c ,ca ,, c ,bb,,b c , ,, , ,b bccc ,a,b,,a, ac ,c ,c c a,a , cc, c, aa c ,bc ,c,c , , ac a ,bc,cc ,, , ac,a , ,cc c aa b, caa b c ,b ,,aa a ,, b , a, ac ba c , , c bb,,b , , ,,aacb b , , ccc , b c,, ,a c , ,a a, a, , a c, , ,a a , , b a, a ,,c,a , b ,b a ,c c, b aa a ,b,,, , a,b a,c, , a c c,b , , a c, ,,,c baba a ,a b ,b , c a , ,b,b , c b,c c,,b,c cb,, cbb b,,, a a , c, ccb, a , , b cb cc c , , abcba,b bbb, c b c a b,a a ,bb,cca ,ab ,,a ,, a cbbb, ba,a , caca a c ,, b , , ca , cb, , c , aa acaa a c ,a a c, b,, ac aaaa , c a b, , c ,,a ,b b a, b ,, bccaa , ,a, , ca, a , a cc, ,b,a,,cacc b bc b,, , ,c aa,aa a, c , c , , bcab, b, a a b a c,, b abcaa,,bac,c, , , b, c a , aba ,acc, ac bba b ,,a b,, a a ,c ,c a,bba a,b b , ,a, b, , c c ,caa,,, c b cc cab c accaa cca a b c b,, a , , ab c , c c b c , c ,cc ,b b,,, b , , c a, ,, b c, , a,,,, ,a,cc c ,b ,a ,, bb, , a b ab cb, c , , ,, bbcbc, , a b , a, b c , , ,b c a ,aa , , ,,, bb a a ,bbc, a a b , a ,, cc ,b b, ,,b bcb, a a, , ,, cb ,a b , , , b, , c,c c , , , c aaa, b aa, b, bcb b c ba a c c a,a a, c c ,ca a ba, ab, bb b,, c ca b a a , a , ,a ,,c c, , ,a b cb,cb,,b, c ,,, a, b, a , aa, c,b,, c,a ,bcb , cb,a c ,c a b,a,b, , b a b, ba,a, b,,c, ac,ab ,aacc ,a bb , aa caa b ,, ,b a ,b, ,c, a, ,a,c , b c a , , ,,ca b ac , b ,, ,,b , , ,, ,b, a aaa,ba ,cc ,,a , , ,c , c, , cb,c a ,, c a c , b, cca, b , b,,c c c ,b,, ,b a,,b, , , b,bc c ,, c, bb ,a, , ,, a cb c, b ,,b ,,,c c cc , b ,a,a c c ,,b ,,a a,ac c,b , , ,b ca, ab,,cc c cab c ,,a, , ba, ,,,c b, ,c , b bc , a a, c, b c a bc , bb,cb , a c c aa c aa, c a c, , b ,b cac bcc , b , ,,c ,aa ,,a, ,c b ab ,c ,a a, ,c b cab b a c ,a ,cacb ab , , ,ac a b c , abaab c,, ,cc aa b a, ,b a a,c, , c,c c, b ca,c ba,b ,cba, a ab,, ,a ,c b b b ,c b, ,, c,aab bb,a,aa b a ,b b, cc b , a ,a b,,cc ,aa c , ab c b b bbb ,,, c ba a bcca ac , b, , b,, c ca cb , a ,,b,,a,, ,bc a a b ,,, , b,c ,c c, ,b bc aac , b a ,a b ca , b, b , c cc c caba,aa, cb ac c b b,,bb b c a , a, baba c ,a b c b,, ,c b,,a a bb c ,,a,,a , a ,, ,b ,,, b,,cbccc,c a c,b ,,c b b, b , a , b c bba b,cb c b , ,, ,c a cba bc, , c, b baa b , c ,, ,,, ,,, a ,a ,,c , ab ab b, , c aa , c , a ,a a c ,, c b, , , a b ,, b , ,, a b bc b , a aa , c a c , b, bcc b , , a bba ac a a ,, b a, ac, b , bcac,,,c c b,b aa , ,a ,,, , b , a,b bb,baa ,,b c bc,, a b b ,, , , ac ,,, ,b b,c b, ac ac,, , , ,b ,ba, cb, c c, aca,, ca ,b ,aac a,,c acb , a,, a ab,bc ab ba,a, , ,c,, ac,,,ac, ab,, a, b,a , ccc ,b, a,a ba,, , ,c , cb ac , ,a, , ca , , a,b, ,a a c , b,a, cba , , ,,, a a,, a a c c c, c , bac ,, , b b c , c, aa ,,, ,, bb a ,b, , ,, b,c,,a ,b c a, c , , a b a,a,ab b b, a ,c ,b,cb , b, cbc , ,c,ca cba, bc , b,bc c a, c, , a, b, , a , b ,c ,b ,cc ,ac , b a ab c b , , c , b aa c b , c c , , , , a, , c , b bc,cb, c a b ,b bca , cc cc c , c c a , , c b , bbb ,, ,abb b ca c , ,, ba,a a b ba cca, a,, aab a bab ,a , cb, ,b a c,,bc caac b, , ,,c ba ,a, b,, , ,, ,a , acb cccb , , c, bcc b,c ca, b b , ,, a,, b,,c a , , a, a a , a a , b a, , , , a c b ac,c , c , bb, b,b ,ac, cc ba, c , , b, , , c,b ,, b cb , ba cbaa a b c,,cc b ,, bc b,, , a cb,aa abc aa, b,ca, b a,, , c,ac ,b c ccac, aabc a, ca a ,,,,b a ,c b aca c , c, a a b,c, , c a c b , c ,, b, a, c , c , , b, a,,b,ab, ,a ,b,, a bac ,b b ,c c ,,, c , , bb a bcb , bb, c , bc a ,, ,cc , ,, ,,b, bb b ,b,c c c , , , a, , ,ba,a c, c, c b b a b , , , c ,c,, cb b,c ba bc a, ca ,bc aa , ,cb ,,bbc, , , , a a cbb, c, ,, ,c b b ,a, c, c b cb,c, c acc ,b,cc , , ,c , bc,cab , , ,c ,a b, , c, , , c ,,a,, c,b,ca c c ,,a, c, c b ,cbbaa b,bb c ,ab,a,a a c, c, c, a bb, c b, ,, a,c,,,bbb ,b c c c c, c , bc, aa c ab, , c cb ba,b, , a a,c , b bb b a bc ,,,, ,a b a,a ,c b ,c ,b,c a ,abaa a ac , c,b a, a ,,a , bc b c ,, c c ab ,b, c ,b ac , , a ,c b , b bc a b b b c,a, , b, ca b , b,b , ,,a a b a ,, ba , , , ,ba c b a ,c , a , bb c a a c ,, , ca , a a c ,b,bc b, ,bba ,a,c acaa b a ca ,bbbbb ,b,,,caa ,b a bb c, c a b c b,b b ,a ,, , , b b a, c c a,a acb , ,c , b ,,,c,b,,,a c a, ,,c a,,cb , b abbac, , ca b c , a,, b, ,,,bb,b, ,c b ac, a c ,a b, bbc, ab c, , , c ,, , bc ,ab ,bab b , a cb b a b , a ,, ,b,,,,,b a , ,aba ,a b ac , a,c , , ca b , ,ca a , , a,aba a, a aa, cac, ca ,c a,, , caca abb bc c a b ,, bbc c ,,c ,b,, a ccbaabb b ,b, , a , c c ,, a a cc a,c a ,, ,cacb ,a abb c ,b c, ca,a b , ca c b a bb,, ab,ba,c,c b , abb b ba , ,, ,a ,a , c, a c,, ,b ,,b , , ac ,, bcbbb , a cca bb ,b ,c,abc , a ,, a ,bbaa , b, ,c b cc b aa , c bb bc bb,, a a bcac bb c , a, , ,cacc ba cc ,ca bb b ,,, , ,b b ,,c , c , c aca,, , c c, , c a, ,, , , a, c bc b, ,c, ,bb b, ,, b ca a, c a c b caa , c a ,a, , b ,,c , b ab b, ,c , b ,b ,, , b,a ,a b ,a, ,,c bb ca b,cc , a, b aaa, c ,c ,cb,,b b,b, a , a,cca, c ,ccc bba ,,, b , , , b a , b, a, ,, a c b c c ,c,, aa , c c, c ,, ,bc, a bc bb , ba a , b,c ,cb a bb,ac , , ,, ,ca ,bb ba ,b ,c , a ,cb c, , ab,c a c,ab,b ,,b,,, bbb,,a ,, abb,c , b ,, ,a b aa , a ,a, ,ab ca b,a a a, a c cb ba cba , c c , c a , ca ca c a , b ,, , , b ab, ,ba ,, a ,,aaa, ,aaa a c ,,,, cc ,c , ,a b c b a, c c ca c c ,, c , c ba c bb ,, b b a , ,, a, ,bc, ,a a b b ,, b,a,, , ,,, c aa , , ,ab,c c , ,a c aa ,,a, c ba c ,bbca ,, , , c , ,, c, ,c,a ,bccc caca ,,b bac a, c,cc a,, ba b ,c c b ,, abc,ca b ba,,c,, cba, ,c b ca bbbcb b,,bcb cb,, ,,c ,aa aab, , b,,b b , aa,,,c c c ,c , ,acc c ab, c a , , , , c ,a , ,c b,,acc b,b , b c ,ab ,c , b a ,a , a , c a c c, baa , ,, ,b , c, ,b c , , b ,,c,b ,b b a abbc ,, a c c b, b,,b,, b ac ,, , b a, c,,ba,c b,b ,,, ,, b b c,a, a , ,,c c , b ca , , a , a, , ,aaab,, , a , b,ac,, , , , , b,a, c ,a, c,,,, c,,b ,ab, ,b,,, ,cb a ba,a ,,,b ,c, b , , , b c , , a a c aa ba ,a bb a b , , ,a b, b a a,ab,a,, ,,b ,,b, ,, c,,b , , a, , , , ,, ,, a , c, , a,a, a,abb , , ba , aa ,aa,ab , , a,c ,cc, , b , aa , ,a, b a b ,,,,, aa,a,ab a , b, b,c,, , ,,, ba,b b, bb, a a ,a ,c a,c, cbb b , c ,a , , c,,a,b , ,ac ,,,aa a acbca , aa , ,, cb, c,a a, babc b cb, ac,,ca ca,b , a ,a a bc, ,, babc , a bb b ,, ,, b ,ca b,, , a ,, ac, , a ,bc , aa b a a a , c , c bb a cc c,c, , , ,, ab, a,bcc , , bc , ,c a b a, aa, a b ab a,, , b, b, c ,b , , c ,a,cacaac c ,b, c, , a ab b , ,,ca, , caab c a,c,,, b b, ,a ba ,b b ,c a , b a,aab bccbc, a ,,a c ca b ,,a ,bc c ab ,,a ,,, cbcb ab b c ca c cc aaabaac a ,c,,,b , ac c b b b a, ,,b ,,,b b , b ,b bc b b a, c,,c,a,c , a, c,, ac, c cc, bc,, , c bcba ccb ca b ,a c ,ab, a bba,,b , , , ,b a , , c aa ccc a ,c,, a, , abc c ,ca , b, ca, a aa b, ccbcb , c,,,c ,,b,c ,,, ,b, b c, a c b b ,aa c , b , ,c , c, cbb ,,, bb , ,,, aa ,,a ,ba a, a, b ,, b, ab aac b c,a, ccc ,, a ,,b,cca cbb cc, b a, , a b ,c bcc, ac , caacc , b , , , ,b ,ab a c, b, b b c , b cccb bab b,,,, b b , cc , ,, abc , ,, c ,, b c ba c bc ,c, ,, , c,c ,cb bb , b, , b b , , ba ab c ,,,bc bb, , c,bcb,, b,,,, ,, ab ,,b ,,a ,, a ,bc , a ,aa , acbbaa , b a cbb,bca , c aaa c , , c b a , , a aaa,bbcbc ba, ,, b , cb , c ba b c , c , a c , , b b, cb , , bba c , , a , , c,c a, a c ca c c,a , b, ,ac c ,, , acc , ,,ba , bcaa aa b b ca c, abbc aa c,,b b c , , acc b , bb c c ,b c,, , ,a a , ca ,c a ab c, ,,c a c ,b, b , , c,,b , c , ,,a ab, baa ,,a, a , ,,,,ab, b, , ,ba ,c,c,baccc, ,b bc,,,,, ,ca,c , b aa a,, aa a ,c a b,, ,b cc , c b, ab b b,c,a c ,, cc , b b b b , b , ,a,, ,cc b , ,a , c cb , ,b , ,bb cb , , cb c , , ,bc,,, , c, , , ab, ,aa, , b ,bc,a ac , , b , c c, c b,c,a,b, ,, c, , c baa,ac , ,a ,, , a , ab , bba b,,c, cc,,b b ba a , c b c a

**“**

**Celkovy pocet slov je 8423**

**Celkovy pocet slov bez duplicity je 175**

## 3.1 Príklad výpisu meraní:

### 3.1.1 ZIMA - BUBBLE SORT - Testovací retazec 3 :

Text

Description automatically generated

### 3.1.2 LETO BUBBLE SORT - Testovací retazec 3:

Text

Description automatically generated

### 3.1.3 LETO QUICK SORT - Testovací retazec 3:

Text

Description automatically generated

### 3.1.4 LETO RADIX SORT - Testovací retazec 3:

Text

Description automatically generated

## 3.1 Testovacie pomocou duplicitného opakovania funkcií

Tu merám každú funkciu v for cykle v zadefinovanom počte opakovaní : „10, 100, 1000, 10 000, 100 000, 1000 000“ , v programe je to implementované cez konštantu TIMES. Funkcie spracovávajú dáta 10-1000 000 krát.

**Toto testovanie má za efekt znázorniť rýchlosť zbehnutia funkcií a efektivitu pri triedení pola, Tu by sa mali prejaviť nadbytočné volania, ktorých bolo v pôvodnej verzií viditeľne veľa a následne ukázať percentuálny ušetrený čas pri tomto meraní.**

**Testovací retazec:**

**1: „ , a,bb, ac ,ab c, , , ,,, , ccc “**

**2: „ Janka Fero Dano Palo Katka Jakob “**

**3: - Testovací retazec ktorý nájdeme na strane č. 30**

### 3.1.1 ZIMA - BUBBLE SORT - Testovací retazec 1, 2, 3 :

**Pri testovaciom retazci č. 3 dostanem stack overflow.**

|  |  |  |  |
| --- | --- | --- | --- |
| SORTING | Test retazec 1 | Test retazec 2 | Test retazec 3 |
| 10 opakovaní  100 opakovaní  1000 opakovaní  10 000 opakovaní  100 000 opakovaní  1000 000 opakovaní | 0s  0.002s  0.006s  0.06s  0.584s  5,908s | 0s  0.002s  0.012  0.113s  1.134s  11.395s | Nezmeratelné  Nezmeratelné  Nezmeratelné  Nezmeratelné  Nezmeratelné  Nezmeratelné |
| Priemer 1 opakovanie | 0,000005908s | 0,000011395s | Nezmeratelné |
| POLE ORG | Test retazec 1 | Test retazec 2 | Test retazec 3 |
| 10 opakovaní  100 opakovaní  1000 opakovaní  10 000 opakovaní  100 000 opakovaní  1000 000 opakovaní | 0s  0s  0s  0,002s  0.022s  0,216s | 0s  0s  0s  0.005s  0.045s  0.452s | Nezmeratelné  Nezmeratelné  Nezmeratelné  Nezmeratelné  Nezmeratelné  Nezmeratelné |
| Priemer 1 opakovanie | 0,000000216s | 0,000000452s | Nezmeratelné |
| VYMAZ EXTRA MEDZERY | Test retazec 1 | Test retazec 2 | Test retazec 3 |
| 10 opakovaní  100 opakovaní  1000 opakovaní  10 000 opakovaní  100 000 opakovaní  1000 000 opakovaní | 0s  0s  0s  0.001  0.008  0.08s | 0s  0s  0s  0.001s  0.015s  0.155s | Nezmeratelné  Nezmeratelné  Nezmeratelné  Nezmeratelné  Nezmeratelné  Nezmeratelné |
| Priemer 1 opakovanie | 0,00000008s | 0,000000155s | Nezmeratelné |

**Podľa merania jednoznačne vidieť , že veria zo zimy je mnohokrát pomalšia ako tá terajšia z leta.**

### 3.1.2 LETO BUBBLE SORT - Testovací retazec 1, 2, 3 :

|  |  |  |  |
| --- | --- | --- | --- |
| SORTING | Test retazec 1 | Test retazec 2 | Test retazec 3 |
| 10 opakovaní  100 opakovaní  1000 opakovaní  10 000 opakovaní  100 000 opakovaní  1000 000 opakovaní | 0s  0,002s  0,014s  0,001s  0.009s  0.091 s | 0s  0,001s  0,01s  0,001s  0.009s  0.091s | 1,299s  11,751s  nezmeratelné nezmeratelné nezmeratelné nezmeratelné |
| Priemer 1 opakovanie | 0,000000091s | 0,000000091s | 0,11751 |
| POLE ORG | Test retazec 1 | Test retazec 2 | Test retazec 3 |
| 10 opakovaní  100 opakovaní  1000 opakovaní  10 000 opakovaní  100 000 opakovaní  1000 000 opakovaní | 0s  0s  0,001s  0,001s  0.009s  0.085s | 0s  0s  0,001  0,001s  0.009s  0.159 | 0,002s  0,011s  0.131s  nezmeratelné nezmeratelné nezmeratelné |
| Priemer 1 opakovanie | 0,00000085s | 0,00000159s | 0,0000131 |
| VYMAZ EXTRA MEDZERY | Test retazec 1 | Test retazec 2 | Test retazec 3 |
| 10 opakovaní  100 opakovaní  1000 opakovaní  10 000 opakovaní  100 000 opakovaní  1000 000 opakovaní | 0s  0s  0s  0,002s  0.007s 0.079s | 0s  0s  0s  0,002s  0.007s  0.156s | 0,001s  0,014s  0,129s  0.131s  nezmeratelné nezmeratelné |
| Priemer 1 opakovanie | 0,000000079s | 0,000000156s | 0,0000131 |

**Podla Testovania je Bubblesort veľmi efektívny pri menších vstupoch kde je počet slov pod 20 a veľmi nefektívny pri velkých počtoch slov.Priamo až nevyužitelný.**

### 3.1.3 LETO QUICK SORT Testovací retazec 1,2,3 :

Pri test retazci č.3 si zobral max 4200 slov – nefunkčné diskvalifikované

|  |  |  |  |
| --- | --- | --- | --- |
| SORTING | Test retazec 1 | Test retazec 2 | Test retazec 3 |
| 10 opakovaní  100 opakovaní  1000 opakovaní  10 000 opakovaní  100 000 opakovaní  1000 000 opakovaní | 0s  0s  0s  0,001s  0,01s  0,097s | 0s  0s  0s  0,001s  0,01s  0,102s | 0.005  0.047  0.472s  4.702s  43,341s  nezmeratelné |
| Priemer 1 opakovanie | 0,000000097 | 0,000001102 | 0,00043341 |
| POLE ORG | Test retazec 1 | Test retazec 2 | Test retazec 3 |
| 10 opakovaní  100 opakovaní  1000 opakovaní  10 000 opakovaní  100 000 opakovaní  1000 000 opakovaní | 0s  0s  0s  0,001s  0,008s  0,082s | 0s  0s  0s  0,002s  0,016s  0,163s | 0s  0.006  0.057s  0.569s  5,694s |
| Priemer 1 opakovanie | 0,000000082s | 0,0000001503s | 0,00005694 |
| VYMAZ EXTRA MEDZERY | Test retazec 1 | Test retazec 2 | Test retazec 3 |
| 10 opakovaní  100 opakovaní  1000 opakovaní  10 000 opakovaní  100 000 opakovaní  1000 000 opakovaní | 0s  0s  0s  0,001s  0,008s  0,08s | 0s  0s  0s  0,001s  0,017s  0,162s | 0.001  0.006  0.065s  0.654s  6.534 s  nezmeratelné |
| Priemer 1 opakovanie | 0,00000008s | 0,000000156s | 0,00006543 |

**Moja implementácia quicksortu nedáva korektné hodnoty, akonáhle počet slov stúpne viacej ako 50. Tento problém som nevyriešil, ale algoritmus pracuje správne pri testovaciom retazci 1, 2. Tu sa nedeje dodatočné kopírovanie a Quicksort je bleskurýchly.**

### 3.1.4 LETO RADIX SORT - Testovací retazec 1,2,3

|  |  |  |  |
| --- | --- | --- | --- |
| SORTING | Test retazec 1 | Test retazec 2 | Test retazec 3 |
| 10 opakovaní  100 opakovaní  1000 opakovaní  10 000 opakovaní  100 000 opakovaní  1000 000 opakovaní | 0s  0,002s  0,014s  0,138s  1,381s  13,834s | 0s  0,001s  0,01s  0,098s  0,977s  9,774s | 0,006s  0,061s  0,607s  6,067s  60,712s  nezmeratelné |
| Priemer 1 opakovanie | 0,000013834 | 0,000009774s | 0,00060712s |
| POLE ORG | Test retazec 1 | Test retazec 2 | Test retazec 3 |
| 10 opakovaní  100 opakovaní  1000 opakovaní  10 000 opakovaní  100 000 opakovaní  1000 000 opakovaní | 0s  0s  0,001s  0,013s  0,129s  1,4s | 0s  0s  0,002s  0,014s  0,147s  1,503s | 0,002s  0,014s  0,1206s  1,37s  13,728s  nezmeratelné |
| Priemer 1 opakovanie | 0,0000014s | 0,000001503s | 0,00013728s |
| VYMAZ EXTRA MEDZERY | Test retazec 1 | Test retazec 2 | Test retazec 3 |
| 10 opakovaní  100 opakovaní  1000 opakovaní  10 000 opakovaní  100 000 opakovaní  1000 000 opakovaní | 0s  0s  0s  0,001s  0,008s  0,08s | 0s  0s  0s  0,001s  0,015s  0,156s | 0,001s  0,013s  0,129s  1,288s  12,866s  nezmeratelné |
| Priemer 1 opakovanie | 0,00000008s | 0,000000156s | 0,00012866s |

**Podľa pozorovania, algoritmus je neúčinný pri malej dĺžke vstupu a naopak veľmi porovantelný s Quicksortom akonálhe N je dostatočné velké.**

## 3.2 Spriemerované Merania samotného programu

**Tu meráme duplikované spustenie programu cez konštantu TIMES\_WHOLE**

**Quicksort je pri 3 variante diskvalifikovaný**

|  |  |  |  |
| --- | --- | --- | --- |
| BUBBLESORT .ZIMA | Test retazec 1 | Test retazec 2 | Test retazec 3 |
| 10 opakovaní  100 opakovaní  1000 opakovaní  10 000 opakovaní  100 000 opakovaní  1000 000 opakovaní | 0.103s  Nefunguje  Nefunguje  Nefunguje  Nefunguje  Nefunguje | 0.088s  Nefunguje  Nefunguje  Nefunguje  Nefunguje  Nefunguje | Nefunguje  Nefunguje  Nefunguje  Nefunguje  Nefunguje  Nefunguje |
| Priemer | 0,0103 | 0,0088 | Nefunguje |
| BUBBLESORT LETO |  |  |  |
| 10 opakovaní  100 opakovaní  1000 opakovaní  10 000 opakovaní  100 000 opakovaní  1000 000 opakovaní | 0.088s  0.812s  3.255s  14.381s  Nezmeratelné  nezmeratelné | 0.088s  0.912s  3.525s  15.381s  Nezmeratelné  nezmeratelné | 6.612s  Nezmeratelné  Nezmeratelné  Nezmeratelné  Nezmeratelné  Nezmeratelné |
| Priemer | 0,0014381 | 0,0015381 | 0,6612 |
| QUICKSORT |  |  |  |
| 10 opakovaní  100 opakovaní  1000 opakovaní  10 000 opakovaní  100 000 opakovaní  1000 000 opakovaní | 0.087s  0.919s  0.932s  3.23s  15.4s  Nezmeratelné | 0.085s  0.819s  0.903s  3.43s  15.424s  Nezmeratelné | 2.092s  19,203s  Nezmeratelné  Nezmeratelné  Nezmeratelné  Nezmeratelné |
| Priemer | 0,000154 | 0,00015424 | 0,1923 |
| RADIXSORT MSD |  |  |  |
| 10 opakovaní  100 opakovaní  1000 opakovaní  10 000 opakovaní  100 000 opakovaní  1000 000 opakovaní | 0.097s  0.911s  3.253s  15.576  Nezmeratelné  Nezmeratelné | 0.087s  0.911s  3.325s  15.776  Nezmeratelné  Nezmeratelné | 4.514s  37.149s  Nezmeratelné  Nezmeratelné  Nezmeratelné  Nezmeratelné |
| Priemer 1 opakovanie | 0,0015576 | 0,0015776 | 0,37149 |

**Quicksort vyhráva pri Retazci č.1,2. Ale podľa môjho názoru by Radixsort vyhral pri retazci č.3. kde vidieť, že postupne dobieha quicksort.**

# Vývojové diagramy

## 4.1 void vymaz\_extra\_whitespaces(char\* pole\_char)

PSEUDO - FLOWCHART FLOWCHART SO  SYNTAXOM JAZYKA C

Diagram

Description automatically generated

A picture containing graphical user interface

Description automatically generated

## 4.2 FLOWCHART pole\_org

PSEUDO - FLOWCHART FLOWCHART SO  SYNTAXOM JAZYKA C

Diagram

Description automatically generated

Diagram

Description automatically generated

## 4.3 FLOWCHART Bubble sort

PSEUDO - FLOWCHART FLOWCHART SO  SYNTAXOM JAZYKA C

A picture containing diagram

Description automatically generatedA picture containing text, person, screenshot

Description automatically generated

## 4.4 FLOWCHART quickSort

PSEUDO - FLOWCHART FLOWCHART SO  SYNTAXOM JAZYKA C

Diagram

Description automatically generatedDiagram

Description automatically generated

Chart

Description automatically generated

## 4.5 radixSort

PSEUDO - FLOWCHART FLOWCHART SO  SYNTAXOM JAZYKA C

Chart, box and whisker chart

Description automatically generated with medium confidence

Diagram

Description automatically generated

Graphical user interface

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Diagram

Description automatically generated

## 4.6 FLOWCHART vyskyt

PSEUDO - FLOWCHART FLOWCHART SO  SYNTAXOM JAZYKA C

Diagram

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

## 4.7 FLOWCHART vyskyt\_vypis

PSEUDO - FLOWCHART FLOWCHART SO  SYNTAXOM JAZYKA C

Diagram

Description automatically generated

A picture containing graphical user interface

Description automatically generated

## 4.8 FLOWCHART switch\_volba

PSEUDO - FLOWCHART FLOWCHART SO  SYNTAXOM JAZYKA C

Diagram

Description automatically generated

Application

Description automatically generated with low confidence

## 4.9 FLOWCHART void export\_novy\_retazec

Graphical user interface, application, Teams

Description automatically generatedPSEUDO - FLOWCHART FLOWCHART SO  SYNTAXOM JAZYKA C

Diagram

Description automatically generated

Graphical user interface, application

Description automatically generated

Chart, box and whisker chart

Description automatically generatedDiagram

Description automatically generated

## 4.10 FLOWCHART nahr\_interpunkcie

PSEUDO - FLOWCHART FLOWCHART SO  SYNTAXOM JAZYKA C

Graphical user interface, diagram

Description automatically generatedDiagram

Description automatically generated

# Definicia vstupných,pomocných a výstupných premenných:

## 5.1 void vymaz\_extra\_whitespaces - definicia vstupných,pomocných a výstupných premenných

### 5.1.1 Vstupné:

char\* pole\_char (odkázané adresov) – Vstupné pole kde sa uchováva obsah nacitany bud zo suboru alebo priamo od pouzivatela v case 1.

### 5.1.2 Pomocné:

Int i – používa v for cykle ako číslo ktoré zaznamenáva počet iterácií a inkremetuje sa po každom opakovaní. Taktiež obsahuje hodnotu indexu pola.

Int x - obsahuje hodnotu indexu pola.

### 5. 2.3 Výstupné:

char\* pole\_char (odkázané adresov) – Vstupné pole sa prepíše a bude obsahovať upravene hodnoty bez extra medzier.

## 5.2 int pole\_org - definicia vstupných,pomocných a výstupných premenných

### 5.2.1 Vstupné:

char\* pole\_char (odkázané adresov) – Vstupné pole kde sa uchováva obsah nacitany bud zo suboru alebo priamo od pouzivatela v case 1.

### 5.2.2 Pomocné:

Int i – používa v for cykle ako číslo ktoré zaznamenáva počet iterácií a inkremetuje sa po každom opakovaní. Taktiež obsahuje hodnotu indexu pola. – hodnota dvroz -riadok

Int j - obsahuje hodnotu indexu pola – jednorozmerne pole

Int k - obsahuje hodnotu indexu pola – hodnota dvroz - stlpec

Int p - obsahuje hodnotu indexu pola.

### 5.2.3 Výstupné:

char dvroz[MAX\_SKEN\_ZAPIS][MAX\_SKEN] – Organizované pole „slov“

int count\_medzery – počet medzier

## 5.3 void Radixsort - definicia vstupných,pomocných a výstupných premenných

### 5.3.1 Vstupné:

char \*\*pole\_sort – Vstupné pole kde sa uchováva obsah

### 5.3.2 Pomocné:

Int i – používa v for cykle ako číslo ktoré zaznamenáva počet iterácií a inkremetuje sa po každom opakovaní. Taktiež obsahuje hodnotu indexu pola.

int count[256] - počet reťazcov s daným znakom na pozícii k.

int mode; - najčastejší charakter na pozici k

char \*\*bucket[256]; - pomocný bucket na ukladanie výstupu.

char \*\*top[256]; - prvý nepoužitý index v tomto bloku znakov.

### 5.3.3 Výstupné:

char \*\*pole\_sort – Vstupné pole kde sa uchováva utriedený obsah

## 5.4 void Bubblesort - definicia vstupných,pomocných a výstupných premenných

### 5.4.1 Vstupné:

int space\_count - počet medzier

char dvroz[MAX\_SKEN\_ZAPIS][MAX\_SKEN] – Organizované pole „slov“

### 5.4. 2 Pomocné:

int i - používa v for cykle ako číslo ktoré zaznamenáva počet iterácií a inkremetuje sa po každom opakovaní. Taktiež obsahuje hodnotu indexu pola.

int j - obsahuje hodnotu indexu pola.

char tmp – termporary pamäť potrebná pri kopírovaní.

### 5.4.3 Výstupné:

char dvroz[MAX\_SKEN\_ZAPIS][MAX\_SKEN] – Organizované a utriedené pole „slov“

## 5.5 void quickSort - definicia vstupných,pomocných a výstupných premenných

### 5.5.1 Vstupné:

char dvroz[MAX\_SKEN\_ZAPIS][MAX\_SKEN] – Organizované pole „slov“

int lava minimalny pocet slov = 0

int prava – max pocet slov

### 5.5.2 Pomocné:

int i - používa v podmienke while cyklu, obsahuje hodnotu indexu pola.

int j - obsahuje hodnotu indexu pola.

int x = polovica z pravej strany.

char pom – termporary pamäť potrebná pri kopírovaní.

### 5.5.3 Výstupné:

char dvroz[MAX\_SKEN\_ZAPIS][MAX\_SKEN] – Organizované utriedené pole „slov“

## 5.6 int vyskyt - definicia vstupných,pomocných a výstupných premenných

### 5.6.1 Vstupné:

char dvroz[MAX\_SKEN\_ZAPIS][MAX\_SKEN] – Organizované pole „slov“

### 5.6.2 Pomocné:

Int i – používa v for cykle ako číslo ktoré zaznamenáva počet iterácií a inkremetuje sa po každom opakovaní. Taktiež obsahuje hodnotu indexu pola.

int j - obsahuje hodnotu indexu pola.

int k - obsahuje hodnotu indexu pola.

### 5.2.3 Výstupné:

int pocet – celkovy pocet slov.

char dvrozvys[MAX\_SKEN\_ZAPIS][MAX\_SKEN] – Organizované utriedené pole „slov“ .

## 5.7 int vyskyt\_vypis - definicia vstupných,pomocných a výstupných premenných

### 5.7.1 Vstupné:

char dvroz[MAX\_SKEN\_ZAPIS][MAX\_SKEN] – Organizované pole „slov“

int space\_count – počet medzier.

int pocitadlo – Celkovy pocet slov

### 5.7.2 Pomocné:

int c -pocitadlo vykytu pre kazde slovo

int prenos – celkový počet slov bez duplicity

int j obsahuje hodnotu indexu pola.

int i obsahuje hodnotu indexu pola.

### 5.7:3 Výstupné:

char dvrozvys[MAX\_SKEN\_ZAPIS][MAX\_SKEN] – pole pre výskyt

## 5.8 int switch\_volba - definicia vstupných,pomocných a výstupných premenných

### 5.8.1 Výstupné:

char vyber\_ako\_char[100]; - vstupny vyberovy retazec

### 5.8.2 Pomocné:

int vyber - vyberova moznost

### 5.8.3 Výstupné:

char vyber\_ako\_char[100]; - výstupny vyberovy retazec

## 5.9 void export\_novy\_retazec - definicia vstupných,pomocných a výstupných premenných

### 5.9.1 .Vstupné:

int space\_pocet - počet medzier.

char \*pole\_char – odkaz na nezosortovane pole

char \*nazsub1 - názov súboru (pole)

char \*nazsub2 - názov súboru (pole)

char dvroz dvojrozmerné sortované pole

### 5.9.2 Pomocné:

Int i – používa v for cykle ako číslo ktoré zaznamenáva počet iterácií a inkremetuje sa po každom opakovaní. Taktiež obsahuje hodnotu indexu pola.

## 5.10 void nahr\_interpunkcie - definicia vstupných,pomocných a výstupných premenných

### 5.10.1 Vstupné:

char \*pole\_char – odkaz na nezosortovane pole

### 5.10.2 Pomocné:

Int i – používa v for cykle ako číslo ktoré zaznamenáva počet iterácií a inkremetuje sa po každom opakovaní. Taktiež obsahuje hodnotu indexu pola.

int j používa v for cykle ako číslo ktoré zaznamenáva počet iterácií a inkremetuje sa po každom opakovaní. Taktiež obsahuje hodnotu indexu pola.

int k - obsahuje hodnotu indexu pola.

int ch – testovany char

int ozn - testovany char

### 5.10.3 Výstupné:

char \*pole\_char – odkaz na nezosortovane pole bez dodatočnej interpunkcie.

# Podmienky vstupných, výstupných a pomocných premenných:

## 6.1 void vymaz\_extra\_whitespaces - Podmienky vstupných, výstupných a pomocných premenných

Vstupné:

pole\_char[ ] Є char : ASCII, obsahuje pole znakov načinané zo vstupu.

Pomocné:

i Є Z+ vrátane nuly typu int

x Є Z+ vrátane nuly typu int

Výstupné:

pole\_char[ ] Є char : ASCII, obsahuje pole znakov načinané zo vstupu.

## 6.2 int pole\_org - Podmienky vstupných, výstupných a pomocných premenných

Vstupné:

pole\_char[ ] Є char : ASCII, obsahuje pole znakov načinané zo vstupu.

Pomocné:

I Є Z+ vrátane nuly typu int používa v for cykle, obsahuje hodnotu indexu pola

j Є Z+ vrátane nuly typu int , - obsahuje hodnotu indexu pola – jednorozmerne pole

k Є Z+ vrátane nuly typu int, obsahuje hodnotu indexu pola – hodnota dvroz - stlpec

p Є Z+ vrátane nuly typu int , obsahuje hodnotu indexu pola.

Výstupné:

dvroz[ ] [ ] ] Є char : ASCII, obsahuje pole znakov ktoré obsahuje riadky a stlpce, riadky reprezentuju jednotlive slova a stlpce reprezentuju znaky z ktorých pozostávajú slová (riadky).

## 6:3 void Radixsort - Podmienky vstupných, výstupných a pomocných premenných

Vstupné:

pole\_sort[ ] Є char : ASCII, obsahuje pole slov načinaných zo vstupu.

Pomocné:

I Є Z+ vrátane nuly typu int používa v for cykle, obsahuje hodnotu indexu pola

Count [ ) Є Z je typu int Obsahuje pocet reťazcov s daným znakom na pozícii k.

int mode; Є Z+ vrátane nuly typu int, najčastejší charakter na pozici k

bucket ] Є char , pomocný bucket na ukladanie výstupu.

top ] Є char , prvý nepoužitý index v tomto bloku znakov.

Výstupné:

pole\_sort[ ] Є char : ASCII, obsahuje zoradené pole slov načinaných zo vstupu.

## 6.4 void Bubblesort - Podmienky vstupných, výstupných a pomocných premenných

Vstupné:

dvroz[ ] [ ] ] Є char : ASCII, obsahuje pole znakov ktoré obsahuje riadky a stlpce, riadky reprezentuju jednotlive slova a stlpce reprezentuju znaky z ktorých pozostávajú slová (riadky).

Space\_count Є Z+ vrátane nuly typu int , obsahuje pocet medzier.

Pomocné:

I Є Z+ vrátane nuly typu int - používa v podmienke while cyklu, obsahuje hodnotu indexu.

J Є Z+ vrátane nuly typu int - obsahuje hodnotu indexu pola

tmp Є char : ASCII termporary pamäť potrebná pri kopírovaní.

Výstupné:

dvroz[ ] [ ] ] Є char : ASCII, obsahuje utriedené pole znakov ktoré obsahuje riadky a stlpce, riadky reprezentuju jednotlive slova a stlpce reprezentuju znaky z ktorých pozostávajú slová (riadky).

## 6.5 void quickSort - Podmienky vstupných, výstupných a pomocných premenných

Vstupné:

dvroz[ ] [ ] ] Є char : ASCII, obsahuje pole znakov ktoré obsahuje riadky a stlpce, riadky reprezentuju jednotlive slova a stlpce reprezentuju znaky z ktorých pozostávajú slová (riadky).

Lava Є Z+ vrátane nuly typu int

Prava Є Z+ vrátane nuly typu int

Pomocné:

I Є Z+ vrátane nuly typu int - používa v podmienke while cyklu, obsahuje hodnotu indexu.

J Є Z+ vrátane nuly typu int - obsahuje hodnotu indexu pola

X Є Z+ vrátane nuly typu int - polovica z pravej strany.

Výstupné:

dvroz[ ] [ ] ] Є char : ASCII, obsahuje utriedené pole znakov ktoré obsahuje riadky a stlpce, riadky reprezentuju jednotlive slova a stlpce reprezentuju znaky z ktorých pozostávajú slová (riadky).

## 6.6 int vyskyt - Podmienky vstupných, výstupných a pomocných premenných

Vstupné:

Dvroz [ ] [ ] ] Є char : ASCII, obsahuje utriedené pole znakov ktoré obsahuje riadky a stlpce, riadky reprezentuju jednotlive slova a stlpce reprezentuju znaky z ktorých pozostávajú slová (riadky).

Pomocné:

i Є Z+ vrátane nuly typu int, používa v for cykle ako číslo ktoré zaznamenáva počet iterácií a inkremetuje sa po každom opakovaní. Taktiež obsahuje hodnotu indexu pola.

j Є Z+ vrátane nuly typu int, obsahuje hodnotu indexu pola.

k Є Z+ vrátane nuly typu int, obsahuje hodnotu indexu pola.

Výstupné:

Pocet Є Z+ vrátane nuly typu int , obsahuje celkovy pocet slov

dvrozvys [ ] [ ] ] Є char : ASCII, obsahuje utriedené pole znakov ktoré obsahuje riadky a stlpce, riadky reprezentuju jednotlive slova a stlpce reprezentuju znaky z ktorých pozostávajú slová (riadky), toto pole sa vyuziva na vypis vyskytu.

## 6.7 int vyskyt\_vypis - Podmienky vstupných, výstupných a pomocných premenných

Vstupné:

Dvroz [ ] [ ] ] Є char : ASCII, obsahuje utriedené pole znakov ktoré obsahuje riadky a stlpce, riadky reprezentuju jednotlive slova a stlpce reprezentuju znaky z ktorých pozostávajú slová (riadky).

Pomocné:

C Є Z+ vrátane nuly typu int -pocitadlo vykytu pre kazde slovo

Prenos Є Z+ vrátane nuly typu int – celkový počet slov bez duplicity

J Є Z+ vrátane nuly typu int obsahuje hodnotu indexu pola.

i Є Z+ vrátane nuly typu int obsahuje hodnotu indexu pola.

Výstupné:

Dvroz [ ] [ ] ] Є char : ASCII, obsahuje utriedené pole znakov ktoré obsahuje riadky a stlpce, riadky reprezentuju jednotlive slova a stlpce reprezentuju znaky z ktorých pozostávajú slová (riadky) posiela sa ako parameter funkcie, nasledne sa využije na export slov v tomto poli..

## 6.8 int switch\_volba - Podmienky vstupných, výstupných a pomocných premenných

Vstupné:

vyber\_ako\_char[ ) Є char : ASCII – pole - vstup od pouzivatela

Pomocné:

vyber je int = {1,2,3}

Výstupné:

vyber\_ako\_char [ ] ) Є char : ASCII – pole - vystup

## 6.9 void export\_novy\_retazec - Podmienky vstupných, výstupných a pomocných premenných

Vstupné:

space\_pocet - Є Z+ vrátane nuly typu int – pocet medzier

pole\_char Є char : ASCII, obsahuje pole zoradenych slov od sortovacieho algoritmu

nazsub1 Є char : ASCII pole pre názov súboru

nazsub2 Є char : ASCII pole pre názov súboru

dvroz

Pomocné:

i Є Z+ vrátane nuly typu int, používa v for cykle ako číslo ktoré zaznamenáva počet iterácií a inkremetuje sa po každom opakovaní. Taktiež obsahuje hodnotu indexu pola.

## 6.10 void nahr\_interpunkcie - Podmienky vstupných, výstupných a pomocných premenných

Vstupné:

pole\_char Є char : ASCII, obsahuje pole slov načinaných zo vstupu.

Pomocné:

i Є Z+ vrátane nuly typu int, používa v for cykle ako číslo ktoré zaznamenáva počet iterácií a inkremetuje sa po každom opakovaní. Taktiež obsahuje hodnotu indexu pola.

J Є Z+ vrátane nuly typu int, používa v for cykle ako číslo ktoré zaznamenáva počet iterácií a inkremetuje sa po každom opakovaní. Taktiež obsahuje hodnotu indexu pola.

Výstupné:

pole\_char Є char : ASCII, obsahuje pole slov bez dodatocnej interpunkcie

# Vyhodnotenie časovej komplexity - Leto

## 9.1 void vymaz\_extra\_whitespaces

Ak máme 500 znakov tak algoritmus vyhodnotí 500 znakov.

**Obsahuje for cyklus, lineárna časová zlozitosť 0(n).**

## 9.2 int pole\_org

Časovo lineárne zoradenie na slová, preukázalo sa aj v testovaní, algoritmus lineárne narastá v čase pri vykonaní ak sa zvyšuje n.

**Obsahuje for cyklus, lineárna časová zlozitosť 0(n).**

## 9.3 void Radixsort

Sortovací algoritmus.M je dĺžka najdlhšieho stringu.

**Najhoršie možné scenário : O(N\*M)**

**Priemerné scenário : O(N+M)**

**Najlepšie možné scenário : O(N)**

## 9.4 void Bubblesort

Sortovací algoritmus.

**Najhoršie možné scenário : O(n^2)**

**Priemerné scenário : O(n^2)**

**Najlepšie možné scenário : O(n^2)**

## 9.5 void quickSort

Sortovací algoritmus

Najhoršie možné scenário : O(n2) S(n)

Priemerné scenário : O(n\*log n)

Najlepšie možné scenário : O(n\*log n

## 9.6 int vyskyt

Tu vidíme dva cyklusy, jeden je vnorený , tieto cyklusy sa vždy vykonávajú podľa počtu medzier. Je známe, že skoro vždy vnorené cyklusy spôsobujú O(n^2) . V mojom prípade je to ale dané počtom medzier, keď je pocet medzier 5 vrchný cyklus sa vykoná 5 krát a spodný 5 krát, ale ked máme pocet medzier 10 tak aj jeden aj druhý cyklus sa vykoná 10 krát. O = n \* n = n – lineárne vziazanie na pocet opakovaní.

**lineárna časová zlozitosť O(n).**

## 9.7 int vyskyt\_vypis

Podobná situácia ako pri výskyte

**lineárna časová zlozitosť O(n).**

## 9.8 int switch\_volba

Funkcia sa vykonáva konštantne.

**konštantná časová zlozitosť O(1).**

## 9.9 void export\_novy\_retazec

* Prepis súboru je O(n).
* Vlozenie do súboru čo môj program nerobí je iO(n + m), m reprezentuje momentalny pocet bytov v súbore.

**Obsahuje for cyklus, lineárna časová zlozitosť 0(n).**

## 9.10 void nahr\_interpunkcie

Lineárne nahradenie interpunkcie-

**Obsahuje while cyklus, lineárna časová zlozitosť 0(n).**

# Vyhodnotenie priestorovej komplexity -Leto

## 10.1 Priestorová zložitosť funk. void vymaz\_extra\_whitespaces

8B (\* char Pole\_char) + 4B(i) + 4B(x)= 16B. 500 Znakov vyhodnotí, uloží sa bud 500 alebo menej.

**Nenavyšujú sa priestorové nároky S(1)**

## 10.2 Priestorová zložitosť funk. int pole\_org

8B(\* char Pole\_char) +Dvojrozmerné pole ktoré je odkázané hodnotou a nie odkazom, tu by mohla byť v budúcnosti dalšia optimalizácia.Velkost je 20 000 x 50 000 = 1000 000 x 8B = 8 000 000B + 4B(i) +4B (j) + 4B (k) + 4B (p) + 4B(COUNT MEDZERY) = 8000 024B.

Polechar sa v tomto prípade prepisuje do dvroz pola, Čo zvyšuje nároky na pamäť. Kopírujeme rovnaké množtvo pamäte ako je aj vstup.

**Navyšujú sa priestorové nároky S(n)**

## 10.3 Priestorová zložitosť funk. void Radixsort

8B (\*\*char pole\_sort) + 4B(int n) + 4B (int k) + 256\*4 = 1 024B (char count) + 256\*8= 2 048B (char \*\*bucket) + 256\*8= 2 048B (char \*\*top) . To celé má počiatočnú velkosť = 3 088 B

Následne kôli rekurzií dostaneme priestorovú zložitosť:

**S (N + M\*B), kde M = dĺžka najdlhšieho reťazca a B = veľkosť radixu**

## 10.4 Priestorová zložitosť funk. void VymenStringy

8B (char \*\*a) + 8B (char \*\*b) +8B (char \*temp)= 24B

Namiestó equivalentu strcopy je toto omnoho viacej efektívnejšie.

**Nenavyšujú sa priestorové nároky S(1)**

## 10.5 Priestorová zložitosť funk. void Bubblesort

4B(int space\_count\_prev) + 8000 000B (char dvroz) + 4B (int i) + 4B (int j)= 8000 012B

Potrebujeme iba zvýšený priestor na temp.

**Nenavyšujú sa priestorové nároky S(1)**

## 10.6 Priestorová zložitosť funk. void quickSort

4B(int i) + 4B (int j)+ 8000B(char dvroz)+ 4B (int lava)+ 4B (int prava)=8000 016 B

Quicksort je algoritmus, ktorý má viaceró verzií ale v mojej rekurzívnej verzií je najhoršia priestorova zlozistost :

**S(log n)**

## 10.7 Priestorová zložitosť funk. int vyskyt

4B(int space count) + 8 000 000B (char dvroz) + 8 000 000B (char dvrozvys) + 4B (int pocet) + 4B (int i ) + 4B (int j) + 4B (int k) = 16 000 020B

tento počet bytov môže mať na svedomí pretečenie stacku, opreože podľa dalšieho hladania som zistil že stack má velkosť defaultne alokovanú 8,388,608 Bytov. Duplikujeme n čo znamená 2\*n = n

**Navyšujú sa priestorové nároky S(n)**

## 10.8 Priestorová zložitosť funk. int vyskyt\_vypis

4B(int space count) + 8 000 000B (char dvroz) + 8 000 000B (char dvrozvys) + 4B (int pocitadlo) + 4B (int i ) + 4B (int j) + 4B (int c) = 16 000 020B

**Nenavyšujú sa priestorové nároky S(1)**

## 10.9 Priestorová zložitosť funk. int switch\_volba

8\*100 = 800B (char vyxber\_ako\_char) + 4b int vyber = 804B

**Nenavyšujú sa priestorové nároky S(1)**

## 10.10 Priestorová zložitosť funk. void export\_novy\_retazec

4B (int space\_pocet) +8B (char \* pole\_char) + 8B (char \* nazsub1) + 8B (char \* nazsub2) + 8 000 000B (char dvroz) =+ 4B (int i) = 8 000 032 B

**Nenavyšujú sa priestorové nároky S(1)**

## 10.11 Priestorová zložitosť funk. void nahr\_interpunkcie

8B (char \*pole\_char) + 4B (int i) + 4B (int j) + 4B (int k) + 4B (int ch) + 4B (int ozn) + 4B (int ozn)= 28B

**Nenavyšujú sa priestorové nároky S(1)**

# Vlastnosti.

**Vlastnosti algoritmu**

## 11.1 NUTNÉ VLASTNOSTI:

Determinovanosť: Splnená, algoritmus je v každom kroku jednoznačne daný a je určený tok riadenia, Nepoúžívame goto ani iné alternatívy.

Rezultatívnosť: Splnená, algoritmus dáva pre rovnaké vstupy vždy rované výsledky : otestované na testovacích retazcoch, túto vlastnosť by mohlo porušiť pretečenie stacku, ale to v algoritmoch neriešime.

Konečnosť: Splnená, Algoritmus po konečnom počte krokov ,vždy skončí. Algoritmus pozostáva z rôznych typov cyklov ako sú FOR DO-WHILE A WHILE. Každý jednotlivý cyklus je ošetrený a nespôsobí nekonečný loop

## 11.2 ODPORÚČANÉ VLASTNOSTI:

Hromadnosť: Splnená, algorimus pracuje na celej množnej hodnote vstupov, od velkých písmen, malých až interpunkciu a znaky s diakritikou.

Elementárnosť: splnená. Všetky časti algoritmu sú elementárne a nezávisia od

konkrétneho programovacieho jazyka.

Efektívnosť: Splnená, využívame optimálny algoritmus v prípade quicksortu a radixsortu. Dokážeme tým povedať, že efektívne riešime daný problém bez zbytočných volaní.

# Záver

Po konečnej kompárácí programov z LETA aj programu zo ZIMY som dospel k záveru, že radixsort je algoritmus, ktorý dokáže byť veľmi rýcly pre vstupy, ktoré kde je počet znakov n súčastou obrovskej databázy.

Sortovací algoritmus radix je pomalší pri vstupoch kde je počet znakov menší ako 1000. Quicksort je omnoho viacej využitelný algoritmus. Jeho hybridné verzie sa dajú použiť efektívnejšie na danú problematiku.

Ak by som daný problém riešil znovu, využil by som timsort, alebo samotný quicksort ako vstavaná funkciu. Sortovacie algoritmy predstavujú veľmi zaujímave odvetvie programovania a teším sa ak s nimi budem pracovať znovu.

Trebalo by ešte spomenúť, že na danú problematiku môže byť dobrý aj merge sort alebo inserstion sort. Insersion sort je dobrá alternatíva ak sa quicksort rekurzívne zacyklí. Môžu sa skombinovať pre dosiahnutie najlepších výsledkov.

# Zdroje

<https://www.geeksforgeeks.org/quick-sort/>

<https://www.quora.com/How-is-recursion-used-in-quicksort>

<https://en.wikipedia.org/wiki/Quicksortň>

<https://en.wikipedia.org/wiki/Radix_sort>

<https://www.geeksforgeeks.org/radix-sort/>

<https://www.youtube.com/watch?v=nu4gDuFabIM>

<https://www.cs.yale.edu/homes/aspnes/pinewiki/RadixSort.html>

<https://de.acervolima.com/msd-(most-significant-digit)-radix-sort/>

<https://www.geeksforgeeks.org/bubble-sort/>

<https://stackoverflow.com/questions/32543683/radix-sort-array-of-strings>

<https://www.cs.princeton.edu/courses/archive/fall04/cos226/lectures/radix.4up.pdf>

<https://sp18.datastructur.es/materials/lab/lab13/lab13>

<https://www.reddit.com/r/ProgrammerHumor/comments/a1lmcn/this_is_fine/>