

# Algoritmos e Estrutura de Dados I - Trabalho 1 e 2

... e as coisas começam a ficar #divertidas

[Home](#)[Publicações](#)[Apresentações](#)[Projetos](#)[FAQ](#)[Prólogo](#)[A busca da salvação](#)[A entrada de dados](#)[Etapa 1](#)[Etapa 2](#)[Ranking](#)[Compilando o meu programa](#)[Calculando o Score do meu programa](#)[Exemplos de Entrada](#)[Prazos](#)[Critérios da Correção](#)

Que #trabalho #divertido. #GO #AED1

## DUCKTER - AED22CP - 2016/2

Bruno César Ribas

Notória ajuda de Cristian Pastro e Marcus Antunius

[Like](#) Be the first of your friends to like this.

**Atenção** - Este documento ainda se encontra em desenvolvimento. Atualizações serão constantes.

## Prólogo

O mundo não é mais como antigamente. As notícias, agora, são lidas pelo computador e ordenadas de acordo com algum algoritmo que identifica o que é de maior interesse para algum grupo de usuários.

Antigamente as notícias eram lidas em jornais impressos que eram gerados no dia anterior. Em resumo, líamos notícias defasadas mas podíamos decidir por nossa conta qual notícia achávamos mais pertinente para o nosso dia.

O mecanismo de utilização de um algoritmo para a escolha de notícias é muito bacana se não pensarmos muito a respeito, afinal você abre o endereço de notícias e consegue ler o que te interesse, ou ao menos o que você acha que te interessa.

O que percebemos ao longo do tempo é que esses algoritmos tendem a criar uma bolha de notícias. Você acaba ficando preso em uma escolha algorítmica que não permite que você saia. Muitos estudos e até livros foram feitos sobre isso, veja [aqui](#), [aqui também](#), e [outra vez aqui](#).



## A busca da salvação

- Atualizado 04/10 - correção do nome para coincidir no acrônimo

Sabendo de todo esse problema, a empresa mais ética do país - Agrupamento Ético de Dados Duck Duck Computação Pervasiva, ou simplesmente AED22CP, convidou você, um nobre programador e pensador algorítmico para desenvolver uma solução de busca e *tageamento*.

A AED22CP entende muito bem o problema de bolha algorítmica e precisa de sua ajuda para conseguir realizar operações com o grande site de notícias micro-blogging, o duckter.

O duckter possui um volume de milhares de operações por dia. Algumas dessas operações são:

- Adicionar uma CHAVE única com um conteúdo específico;
- Receber um HIT de uma #TAG que aponta para uma chave;
- Buscar as #TAGS e CHAVES mais consultadas e criar uma página de *trending topic*.



Agora vamos detalhar mais o que são as operações que você deverá fazer.

## A entrada de dados

Como dito anteriormente, a AED22CP é muito ética e por isso você não terá acesso ao conteúdo completo do duckter.

O seu programa irá processar as informações ao longo dos dias, e não será encerrado a cada troca de dia, mas receberá a informação de que um dia passou.

Este trabalho foi dividido em duas etapas. A primeira etapa conta como trabalho 1 e segunda etapa como trabalho 2. Um subconjunto de comandos deverá ser implementado como parte do trabalho 1.

A etapa 2 depende dos comandos da etapa 1, ou seja, a etapa 2 é um incremento dos comandos da etapa 1.

A entrada dos dados será feita por meio de comandos. Os comandos são:

### Etapa 1

#### add key CHAVE: conteudo

exemplo:

```
add key teste: o dia tem um belo azul no horizonte
```

- CHAVE: é a chave que será adicionada, máximo de 50 caracteres
- conteúdo: é o conteúdo que a chave está indexando, máximo de 1000 caracteres;
- A quantidade de chaves a serem adicionadas é indeterminada.

#### tag hit #tag chave

- Atualizado 04/10/2016

exemplo:

```
tag hit #belo teste
```

- **#tag:** é o nome da tag que foi colocada em algum post, tamanho máximo de 1000 caracteres
- **chave:** é o código da chave que esta tag referencia. Note que este comando garante que a tag que a *tag hit* aponta para uma chave adicionada previamente.
- A quantidade de tags a serem adicionadas é indeterminada.

#### show tagcontent #tagid

- Este comando busca o conteúdo da chave para qual uma tag aponta, exemplo:

```
add key belo: um belo dia floresce  
tag hit #flor belo
```

```
show tagcontent #flor
```

- A saída deverá ser o conteúdo da chave *belo*, como abaixo:

```
#flor -> belo
belo :. um belo dia floresce
```

## list trending top XX

- Atualizado 07/10 - formato de saída definido
- Calcula as XX% tags com mais hits do dia
- XX é a porcentagem.
  - ex: 10 representa as top 10% (de todas as tags registradas) com mais hits

exemplo:

```
add key b4d2fc29d0c0bcc5b9e3: NEP Stood silence hesitating This of
add key 3ddf96f836812b6c599d: MON To December. with napping, each
add key 61c28c2d271e75094236: BOT Door. no peering, mortal faintly
add key 8e9fc18900fd7a78a2d1: LBA Gently came of door. there
add key d2685fa13696eae7db0f: SAM Silence nothing each entreating whispered
tag hit #d2685fa13696eae7db0f 61c28c2d271e75094236
tag hit #3ddf96f836812b6c599d d2685fa13696eae7db0f
tag hit #b4d2fc29d0c0bcc5b9e3 61c28c2d271e75094236
tag hit #bc92e96068f6e66960de b4d2fc29d0c0bcc5b9e3
tag hit #8e9fc18900fd7a78a2d1 3ddf96f836812b6c599d
tag hit #8e9fc18900fd7a78a2d1 3ddf96f836812b6c599d
tag hit #b4d2fc29d0c0bcc5b9e3 61c28c2d271e75094236
tag hit #d2685fa13696eae7db0f 61c28c2d271e75094236
tag hit #8e9fc18900fd7a78a2d1 3ddf96f836812b6c599d
tag hit #75d31a34e4a3c8e2170e 8e9fc18900fd7a78a2d1
tag hit #8e9fc18900fd7a78a2d1 3ddf96f836812b6c599d
tag hit #d2685fa13696eae7db0f 61c28c2d271e75094236
tag hit #75d31a34e4a3c8e2170e 8e9fc18900fd7a78a2d1
tag hit #cef71e917f46b85dc5 d2685fa13696eae7db0f
tag hit #33d31e1535be3a7cf57c b4d2fc29d0c0bcc5b9e3
tag hit #61c28c2d271e75094236 61c28c2d271e75094236
tag hit #bc92e96068f6e66960de b4d2fc29d0c0bcc5b9e3
tag hit #61c28c2d271e75094236 61c28c2d271e75094236
tag hit #61c28c2d271e75094236 61c28c2d271e75094236
tag hit #75d31a34e4a3c8e2170e 8e9fc18900fd7a78a2d1
tag hit #d2685fa13696eae7db0f 61c28c2d271e75094236
tag hit #2bd93b04442d10c772e8 3ddf96f836812b6c599d
tag hit #8e9fc18900fd7a78a2d1 3ddf96f836812b6c599d
tag hit #bc92e96068f6e66960de b4d2fc29d0c0bcc5b9e3
tag hit #33d31e1535be3a7cf57c b4d2fc29d0c0bcc5b9e3
tag hit #d2685fa13696eae7db0f 61c28c2d271e75094236
tag hit #33d31e1535be3a7cf57c b4d2fc29d0c0bcc5b9e3
tag hit #3ddf96f836812b6c599d d2685fa13696eae7db0f
tag hit #bc92e96068f6e66960de b4d2fc29d0c0bcc5b9e3
tag hit #cef71e917f46b85dc5 d2685fa13696eae7db0f
list trending top 30
```

saída:

```
Begin 30% top trending
1 #8e9fc18900fd7a78a2d1 with 5 hits
1 #d2685fa13696eae7db0f with 5 hits
2 #bc92e96068f6e66960de with 4 hits
End Trending
```

- Note a primeira e última linha que representam o início de fim da impressão do trending
- A ordem de impressão das tags é pela quantidade de hits (maior para menor) e em caso de empate deve ser impresso em ordem lexicográfica
- Para cada tag do trending existirá uma linha, sendo:

- o o primeiro campo a classificação da tag, tags com a mesma quantidade de hits possuem a mesma classificação (e isso não influencia na contagem dos XX% pedidos)
  - Os números da classificação devem ser impressos justificados a esquerda com 3 campos (leia o manual do printf(3))
- o o segundo campo é a tag seguido da palavra 'with' e em seguida a quantidade de hits da tag, finalizado com a palavra hits

## list trending bottom XX

- Atualizado 07/10 - formato de saída definido
- Calcula as XX% tags com menos hits do dia
- XX é a porcentagem.
  - o ex: 10 representa as top 10% (de todas as tags registradas) com mais hits

exemplo:

```
add key b4d2fc29d0c0bcc5b9e3: NEP Stood silence hesitating This of
add key 3ddf96f836812b6c599d: MON To December. with napping, each
add key 61c28c2d271e75094236: BOT Door. no peering, mortal faintly
add key 8e9fc18900fd7a78a2d1: LBA Gently came of door. there
add key d2685fa13696eae7db0f: SAM Silence nothing each entreating whispered
tag hit #d2685fa13696eae7db0f 61c28c2d271e75094236
tag hit #3ddf96f836812b6c599d d2685fa13696eae7db0f
tag hit #b4d2fc29d0c0bcc5b9e3 61c28c2d271e75094236
tag hit #bc92e96068f6e66960de b4d2fc29d0c0bcc5b9e3
tag hit #8e9fc18900fd7a78a2d1 3ddf96f836812b6c599d
tag hit #8e9fc18900fd7a78a2d1 3ddf96f836812b6c599d
tag hit #b4d2fc29d0c0bcc5b9e3 61c28c2d271e75094236
tag hit #d2685fa13696eae7db0f 61c28c2d271e75094236
tag hit #8e9fc18900fd7a78a2d1 3ddf96f836812b6c599d
tag hit #75d31a34e4a3c8e2170e 8e9fc18900fd7a78a2d1
tag hit #8e9fc18900fd7a78a2d1 3ddf96f836812b6c599d
tag hit #d2685fa13696eae7db0f 61c28c2d271e75094236
tag hit #75d31a34e4a3c8e2170e 8e9fc18900fd7a78a2d1
tag hit #cef71e917f46b85dc5 d2685fa13696eae7db0f
tag hit #33d31e1535be3a7cf57c b4d2fc29d0c0bcc5b9e3
tag hit #61c28c2d271e75094236 61c28c2d271e75094236
tag hit #bc92e96068f6e66960de b4d2fc29d0c0bcc5b9e3
tag hit #61c28c2d271e75094236 61c28c2d271e75094236
tag hit #61c28c2d271e75094236 61c28c2d271e75094236
tag hit #75d31a34e4a3c8e2170e 8e9fc18900fd7a78a2d1
tag hit #d2685fa13696eae7db0f 61c28c2d271e75094236
tag hit #2bd93b04442d10c772e8 3ddf96f836812b6c599d
tag hit #8e9fc18900fd7a78a2d1 3ddf96f836812b6c599d
tag hit #bc92e96068f6e66960de b4d2fc29d0c0bcc5b9e3
tag hit #33d31e1535be3a7cf57c b4d2fc29d0c0bcc5b9e3
tag hit #d2685fa13696eae7db0f 61c28c2d271e75094236
tag hit #33d31e1535be3a7cf57c b4d2fc29d0c0bcc5b9e3
tag hit #3ddf96f836812b6c599d d2685fa13696eae7db0f
tag hit #bc92e96068f6e66960de b4d2fc29d0c0bcc5b9e3
tag hit #cef71e917f46b85dc5 d2685fa13696eae7db0f
list trending bottom 40
```

saída:

```
Begin 40% bottom trending
10 #2bd93b04442d10c772e8 with 1 hits
9 #cef71e917f46b85dc5 with 2 hits
9 #b4d2fc29d0c0bcc5b9e3 with 2 hits
9 #3ddf96f836812b6c599d with 2 hits
End Trending
```

- Note a primeira e última linha que representam o início de fim da impressão do trending
- A ordem de impressão das tags é pela quantidade de hits (menor para maior) e em caso de empate deve ser impresso em ordem lexicográfica reversa
- Para cada tag do trending existirá uma linha, sendo:

- o o primeiro campo a classificação da tag, tags com a mesma quantidade de hits possuem a mesma classificação (e isso não influencia na contagem dos XX% pedidos)
  - Os números da classificação devem ser impressos justificados a esquerda com 3 campos (leia o manual do printf(3))
- o o segundo campo é a tag seguido da palavra 'with' e em seguida a quantidade de hits da tag, finalizado com a palavra hits

## dump tags

- Atualizado 07/10 - novo padrão de saída
- Atualizado 04/10 - impressão em ordem lexicográfica
- Este comando indica que seu programa deverá imprimir todas as tags armazenadas na ordem lexicográfica, no formato:

```
#tag -> chave :: hits=%d
```

exemplo:

- entrada

```
add key belo: um belo dia floresce
tag hit #flor belo
tag hit #dia belo
tag hit #dia belo
dump tags
```

- saída

```
8<-----Begin Tag Dump-----
#dia -> belo :: hits=2
#flor -> belo :: hits=1
8<-----End   Tag Dump-----
```

- Fique atento com as linhas delimitadoras (copie e cole ela dentro do seu printf)

## dump keys

- Atualizado 16/11 - Removido linha que sobrava no exemplo
- Atualizado 07/10 - novo padrão de saída
- Atualizado 04/10 - impressão em ordem lexicográfica
- Este comando indica que seu programa deverá imprimir todas as keys armazenadas na ordem lexicográfica, no formato:

```
chave content="conteudo" refs=%d
```

exemplo:

- entrada


```
add key belo: um belo dia floresce
add key orange: is the new black
tag hit #flor belo
tag hit #dia belo
tag hit #dia belo
add key maluco: forever alone
tag hit #chato orange
dump keys
```

- saída

```
8<-----Begin Key Dump-----  
belo content="um belo dia floresce" refs=2  
maluco content="forever alone" refs=0  
orange content="is the new black" refs=1  
8<-----End Key Dump-----
```

- Fique atento com as linhas delimitadoras (copie e cole ela dentro do seu printf)

## new day


-  16/11 - NEW DAY faz parte da etapa 1
- Comando que inicia um novo dia!
- Note que SEMPRE que seu programa inicia deve ser considerado um novo dia sem nenhuma adição
- Quando recebido o comando **new day** o seu programa deverá remover todas as tags que não tiveram hit no dia.

exemplo:

```
add key lindo: um lindo dia comeca hoje  
add key etico: AED22CP eh muito etico  
tag hit #dia lindo  
tag hit #duckter etico  
new day  
tag hit #dia lindo  
new day
```

- no exemplo acima, adiciona-se as chaves *lindo* e *etico*, e depois tem *hit* nas tags *#dia* e *#duckter*, logo ambas as tags possuem 1 *hit* cada.
- com o comando **new day**, as tags ficam com seus hits zerados, mas permanecem armazenadas
- *hit* para a tag *#dia*, ficando com 1 hit
- com o comando **new day**, a tag *#duckter* deve ser removida, pois ficou sem *hit* no dia que passou e a tag *#dia* permanecem armazenada pois teve um *hit*.

## Etapa 2

-  16/11 - Etapa 2 foi **CANCELADA** fica valendo apenas a etapa 1 que teve o new day inserido

## rm key CHAVE

exemplo:

```
rm key teste
```

## rm tag #tag

exemplo:


```
rm tag #belo
```

## rm brokentangref

- Este é um mecanismo para deixar as listas saudáveis. O comando **rm key** pode deixar algumas tags sem referência válida. Se a chave *teste* existe e a tag

*#belo* aponta para a chave *teste*, depois de remover a chave *teste* a tag *#belo* fica sem apontador e com o comando deste item a tag *#belo* deverá ser removida

## new day

-  16/11 - Este comando foi movido para a etapa 1
- Comando que inicia um novo dia!
- Note que SEMPRE que seu programa inicia deve ser considerado um novo dia sem nenhuma adição
- Quando recebido o comando **new day** o seu programa deverá remover todas as tags que não tiveram hit no dia.

exemplo:

```
add key lindo: um lindo dia comece hoje
add key etico: AED22CP eh muito etico
tag hit #dia lindo
tag hit #duckter etico
new day
tag hit #dia lindo
new day
```

- no exemplo acima, adiciona-se as chaves *lindo* e *etico*, e depois tem *hit* nas tags *#dia* e *#duckter*, logo ambas as tags possuem 1 *hit* cada.
- com o comando **new day**, as tags ficam com seus hits zerados, mas permanecem armazenadas
- *hit* para a tag *#dia*, ficando com 1 hit
- com o comando **new day**, a tag *#duckter* deve ser removida, pois ficou sem *hit* no dia que passou e a tag *#dia* permanecem armazenada pois teve um *hit*.


## rm orphankey

- Este comando remove as chaves que não possuem tags as referenciando, exemplo:

```
add key teste: um teste de chave
add key orphan: uma chave forever alone
tag hit #maluco teste
rm orphankey
```

- neste exemplo a chave *orphan* deverá ser removida

## Ranking

-  Nova conta de cálculo do score 15/10
  - e também como compilar e calcular o score
- Atualizado - 10/10

O ranking será feito por disputas diárias. Para participar basta enviar o código que o sistema compilará e executará.

Para participar do Ranking o seu código deve funcionar no UBUNTU dos laboratórios. Se o sistema não conseguir compilar seu programa apenas será avisado que não conseguiu e nada mais.

A submissão para o Rank é **Obrigatória**.

No Rank a ordenação será considerada pelo SCORE, definido como:

$$((\text{Tempo de Execução em segundos}) * 100 + (\text{Memória Usada em MegaBytes}) * 10) / 110$$

- Quanto menor, melhor.

- A submissão para o sistema de rank iniciará ao menos 2 semanas antes da entrega do trabalho;
- As submissões **não** serão executadas imediatamente. O sistema fará algumas execuções diárias. A quantidade de submissões será definida no momento da abertura das submissões;

## Compilando o meu programa

Para a execução dos testes as soluções serão compiladas com as seguintes FLAGS:

```
-static -O2
```

Exemplo:

```
gcc -static -O2 simples.c -o simples
```

## Calculando o Score do meu programa

Para calcular o score do seu programa instale o pacote "time" (nas máquinas do DAInf ele já está instalado), e execute o seu programa da seguinte forma:

```
/usr/bin/time -f "%M %e" ./meuprograma < arquivo-de-entrada > resposta-do-meuprograma
```



Exemplo:

```
$ /usr/bin/time -f "%M %e" ./simples < 05-sample5-hugeforasample.in > 05-sample5-hugeforasample.sol
2520 1.64
```

O primeiro número que saiu impresso é a quantidade de memória em KBytes e o segundo número é o tempo em segundos. Logo o score para esse exemplo é:

$$(1.64 \cdot 100 + (2520/1024) \cdot 10) / 110 = 1.71$$

## Exemplos de Entrada


-  16/11 - Adicionado primeiros exeomplos com newday
-  16/11 - Adicionada saídas esperadas
- Entradas de 03 a 07 atualizadas
- Criado - 10/10
- [01-sample1-verysmall.in](#) [01-sample1-verysmall.out](#)
- [02-sample2-verysmall.in](#) [02-sample2-verysmall.out](#)
- [03-sample3-small.in](#) [03-sample3-small.out](#)
- [04-sample4-smallstress.in](#) [04-sample4-smallstress.out](#)
- [05-sample5-hugeforasample.in](#) [05-sample5-hugeforasample.out](#)
- [06-sample6-crazytrending.in](#) [06-sample6-crazytrending.out](#)
- [07-sample7-crazytrending2.in](#) [07-sample7-crazytrending2.out](#)
- [08-sample8-newday-rises.in](#)
- [09-sample9-newday-twodays.in](#)
- [10-sample10-newday-craydays.in](#)

## Prazos

- Para a ETAPA 1: ~~6 13 20 de Novembro~~ **11 de Dezembro** às 23h59
- Para a ETAPA 2: ~~10 11 de Dezembro~~ às 23h59

## Critérios da Correção



-  16/11 - Atualizações de acordo com o combinado em sala
- ~~Cada etapa vale 100 pontos~~
- O trabalho vale 200 pontos
- Os trabalhos serão executados contra todas as entradas publicadas no rank.
  - O rank definirá 2 limiares definidos como "*Simples*" e "*Super Simples*"
    - Ficando abaixo do "*Super Simples*" o trabalho valerá no máximo 50 pontos;
    - Ficando entre o "*Simples*" e o "*Super Simples*" o trabalho valerá no máximo ~~80~~ 130 pontos;
  - Se o programa falhar em qualquer entrada terá um desconto de 10 pontos, e desconto adicional de 5 pontos por cada entrada que falhar.
    - Se falhar em apenas 1 entrada terá um desconto de 10 pontos, se falhar em 2 entradas o desconto será de 15 pontos, em 3 entradas desconto de 20 pontos, e assim por diante.
  - Se o programa falhar em 50% ou mais das entradas, terá nota automaticamente em 0.
- Os programas que não falharem em nenhuma entrada entrarão na rodada especial para ponto extra
  - Na rodada especial será executado uma nova entrada
  - Se o programa falhar na rodada especial terá um desconto de 10 pontos
  - Os 3 melhores programas receberão 10 pontos extras.
- Além da rodada automática os trabalhos passarão pelas seguintes avaliações:
  - Será considerado o código:
    - Otimização
    - Limpeza do código
    - Criatividade
  - ~~Para o aluno ou dupla que não fizerem a etapa 1 no prazo, mas concluírem a etapa 2, receberão no máximo nota 60 para etapa 1;~~
  - Defesa individual do código quando a  $T1/2 - P3 \geq 40$  para etapa 1, e
  - ~~Defesa individual do código quando a  $T2 - P3 \geq 40$  para etapa 2, e~~
    - Lembrando que quando o trabalho for em dupla apresenta quem tiver a maior diferença

---

Last Modified: Wed Nov 16 21:36:42 2016. [Bruno César Ribas](https://www.brunoribas.com.br/aed1/2016-2/trabalho1/)