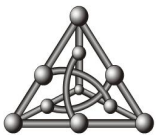


UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
Faculdade de Computação

Implementação Algorítmica

Atividade 2 — Algoritmos de ordenação

1 Descrição

Ordenação é uma operação básica em Computação. Diversos algoritmos de ordenação têm sido propostos ao longo da história, mesmo quando computadores ainda não existiam. A partir de meados do século passado, os algoritmos começaram a ser transformados em programas e executados em computadores. Esta atividade pede que você implemente os algoritmos de ordenação que estudamos e realize experimentos com os mesmos.

Em particular, você deve implementar os seguintes algoritmos de ordenação: BUBBLESORT, SELECTIONSORT, INSERTIONSORT, MERGESORT, HEAPSORT, QUICKSORT, COUNTINGSORT, RADIXSORT e BUCKETSORT.

Dessa forma, você deve implementar 9 algoritmos de ordenação listados acima.

2 Programa, entrada e saída

Considere os seguintes parâmetros para execução dos experimentos, que são fornecidos como entrada:

- **`inc`** é o tamanho inicial de um vetor de entrada;
- **`fim`** é o tamanho final;
- **`stp`** é o intervalo entre dois tamanhos; e
- **`rpt`** é o número de repetições a serem realizadas.

Para realizar experimentos com esses algoritmos, você deve construir 4 tipos de conjuntos de dados de entrada, conforme a descrição a seguir:

1. **Vetor aleatório**: cada conjunto de entrada A deve conter números inteiros não negativos escolhidos (pseudo)aleatoriamente. Um tal conjunto de números A deve ter n números. Os valores de n variam, para cada conjunto construído, de acordo com os parâmetros **`inc`** (valor inicial), **`fim`** (valor final) e **`stp`** (intervalo entre dois tamanhos). Por exemplo, se **`inc`** $= 100$, **`fim`** $= 1000$ e **`stp`** $= 10$, então os tamanhos dos vetores de entrada A que devem ser construídos são $n = 100, 110, 120, \ldots, 990, 1000$. Suponha que o valor máximo possível é 20000.

   Um conjunto A com n elementos assim gerado é chamado de **caso de teste**. Para cada caso de teste, você deve executar os nove algoritmos mencionados na seção 1 para ordenar esse

conjunto A de n números inteiros. Para cada n, você deve repetir o processo acima um determinado número **`rpt`** de vezes, obtendo então a média dos tempos medidos.

A saída deste experimento consiste de uma primeira linha contendo o rótulo **`[[RANDOM]]`**, especificando o conjunto de dados sendo usado, uma segunda linha contendo os rótulos da quantidade de elementos n e a identificação de cada um dos algoritmos. Cada linha a seguir contém o valor da quantidade de elementos n de um caso de teste e as médias dos tempos gastos da execução de cada algoritmo.

2. **Vetor reverso**: cada conjunto de entrada A deve conter números inteiros não negativos escolhidos (pseudo)aleatoriamente e arranjados em ordem decrescente. Um tal conjunto de números A deve ter n números. Os valores de n variam, para cada conjunto construído, de acordo com os parâmetros **`inc`** (valor inicial), **`fim`** (valor final) e **`stp`** (intervalo entre dois tamanhos). Por exemplo, se **`inc`** $= 10$, **`fim`** $= 100$ e **`stp`** $= 5$, então os tamanhos dos vetores de entrada A que devem ser construídos são $n = 10, 15, 20, \ldots, 95, 100$. Suponha que o valor máximo possível é 20000.

   Para cada caso de teste, você deve executar os nove algoritmos mencionados na seção 1 para ordenar esse conjunto A de n números inteiros. Neste caso, não é necessário repetir a execução dos algoritmos, isto é, cada caso de teste é executado uma única vez, obtendo então os tempos medidos.

   A saída deste experimento consiste de uma primeira linha contendo o rótulo **`[[REVERSE]]`**, especificando o conjunto de dados sendo usado, uma segunda linha contendo os rótulos da quantidade de elementos n e a identificação de cada um dos algoritmos. Cada linha a seguir contém o valor da quantidade de elementos n de um caso de teste e os tempos gastos da execução de cada algoritmo.

3. **Vetor ordenado**: cada conjunto de entrada A deve conter números inteiros não negativos escolhidos (pseudo)aleatoriamente e arranjados em ordem crescente. Um tal conjunto de números A deve ter n números. Os valores de n variam, para cada conjunto construído, de acordo com os parâmetros **`inc`** (valor inicial), **`fim`** (valor final) e **`stp`** (intervalo entre dois tamanhos). Por exemplo, se **`inc`** $= 1000$, **`fim`** $= 10000$ e **`stp`** $= 1000$, então os tamanhos dos vetores de entrada A que devem ser construídos são $n = 1000, 2000, 3000, \ldots, 9000, 10000$. Suponha que o valor máximo possível é 20000.

   Para cada caso de teste, você deve executar os nove algoritmos mencionados na seção 1 para ordenar esse conjunto A de n números inteiros. Neste caso, não é necessário repetir a execução dos algoritmos, isto é, cada caso de teste é executado uma única vez, obtendo então os tempos medidos.

   A saída deste experimento consiste de uma primeira linha contendo um rótulo **`[[SORTED]]`**, especificando o conjunto de dados sendo usado, uma segunda linha contendo os rótulos da quantidade de elementos n e a identificação de cada um dos algoritmos. Cada linha a seguir contém o valor da quantidade de elementos n de um caso de teste e os tempos gastos da execução de cada algoritmo.

4. **Vetor quase ordenado**: cada conjunto de entrada A deve conter números inteiros não negativos escolhidos (pseudo)aleatoriamente e "quase" ordenado crescentemente. Para obter

um vetor dessa forma, você deve arranjar o vetor crescentemente, escolher 10% dos seus elementos e então embaralhá-los. Um tal conjunto de números A deve ter n números. Os valores de n variam, para cada conjunto construído, de acordo com os parâmetros **inc** (valor inicial), **fim** (valor final) e **stp** (intervalo entre dois tamanhos). Por exemplo, se **inc** $= 200$, **fim** $= 20000$ e **stp** $= 200$, então os tamanhos dos vetores de entrada A que devem ser construídos são $n = 200, 400, 600, \ldots, 19800, 20000$. Suponha que o valor máximo possível é 20000.

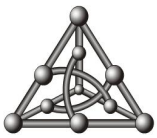
Para cada caso de teste, você deve executar os nove algoritmos mencionados na seção 1 para ordenar esse conjunto A de n números inteiros. Neste caso, não é necessário repetir a execução dos algoritmos, isto é, cada caso de teste é executado uma única vez, obtendo então os tempos medidos.

A saída deste experimento consiste de uma primeira linha contendo o rótulo **[[NEARLY SORTED]]**, especificando o conjunto de dados sendo usado, uma segunda linha contendo os rótulos da quantidade de elementos n e a identificação de cada um dos algoritmos. Cada linha a seguir contém o valor da quantidade de elementos n de um caso de teste e os tempos gastos da execução de cada algoritmo.

2.1 Exemplo de entrada e saída

Um exemplo de execução, para casos de teste com n variando de acordo com os parâmetros **inc** $= 1000$, **fim** $= 20000$ e **stp** $= 1000$, é mostrado a seguir. O parâmetro relativo ao número de repetições foi adotado como **rpt** $= 10$. As médias dos tempos de execução (no caso do experimento aleatório) e os tempos de execução (no caso dos outros experimento) dos algoritmos são mostrados em segundos.

```
[[RANDOM]]
     n       Bubble    Selection    Insertion       Merge        Heap        Quick     Counting       Radix      Bucket
-------------------------------------------------------------------------------------------------------------------------
  1000     0.003200     0.001648     0.000867    0.000355    0.000251    0.000142    0.000154    0.000489    0.000108
  2000     0.014404     0.006451     0.003742    0.000704    0.000543    0.000307    0.000173    0.001039    0.000230
  3000     0.031145     0.014062     0.008867    0.001068    0.000889    0.000497    0.000200    0.001522    0.000398
  4000     0.062955     0.024773     0.014210    0.001417    0.001202    0.000632    0.000187    0.001997    0.000536
  5000     0.086435     0.036523     0.019261    0.001696    0.001481    0.000760    0.000176    0.002260    0.000576
  6000     0.128167     0.053627     0.028429    0.002125    0.001833    0.000971    0.000200    0.002802    0.000725
  7000     0.174926     0.071756     0.039799    0.002662    0.002109    0.001123    0.000213    0.003323    0.000838
  8000     0.244257     0.097852     0.055106    0.002919    0.002497    0.001330    0.000232    0.003853    0.001022
  9000     0.292820     0.118310     0.062595    0.003262    0.002774    0.001458    0.000224    0.004677    0.001334
 10000     0.366109     0.145138     0.077926    0.003651    0.003115    0.001634    0.000233    0.004549    0.001185
 11000     0.444400     0.177583     0.092671    0.004041    0.003466    0.001844    0.000246    0.004974    0.001315
 12000     0.533893     0.209199     0.111599    0.004439    0.003772    0.001964    0.000257    0.005438    0.001439
 13000     0.629903     0.247927     0.130754    0.004853    0.004133    0.002160    0.000266    0.006465    0.001575
 14000     0.737356     0.284025     0.149913    0.005245    0.004471    0.002335    0.000277    0.006323    0.001684
 15000     0.867285     0.337050     0.186726    0.005546    0.005015    0.002650    0.000323    0.007244    0.001929
 16000     0.969104     0.371297     0.196613    0.005649    0.005136    0.002724    0.000307    0.007284    0.001953
 17000     1.094300     0.419200     0.220902    0.006005    0.005502    0.002886    0.000312    0.007695    0.002061
 18000     1.230518     0.471580     0.247990    0.006395    0.005867    0.003059    0.000326    0.008137    0.002178
 19000     1.376001     0.524344     0.277639    0.006790    0.006253    0.003288    0.000336    0.008637    0.002301
 20000     1.531736     0.580426     0.305221    0.007105    0.006577    0.003452    0.000345    0.009038    0.002394
```



[[REVERSE]]

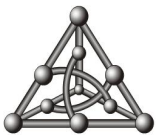
n	Bubble	Selection	Insertion	Merge	Heap	Quick	Counting	Radix	Bucket
1000	0.003568	0.001439	0.001530	0.000278	0.000194	0.000106	0.000133	0.000452	0.000105
2000	0.015544	0.005957	0.006167	0.000587	0.000502	0.000214	0.000160	0.000903	0.000232
3000	0.035119	0.013354	0.013821	0.000842	0.000703	0.000357	0.000177	0.001352	0.000365
4000	0.059738	0.023830	0.024455	0.001154	0.000939	0.000478	0.000163	0.001826	0.000488
5000	0.092080	0.037658	0.038132	0.001424	0.001201	0.000604	0.000173	0.002248	0.000615
6000	0.131257	0.054800	0.055799	0.001731	0.001467	0.000763	0.000184	0.002737	0.000728
7000	0.186980	0.075965	0.076569	0.002014	0.001745	0.000830	0.000193	0.003155	0.000836
8000	0.244005	0.103281	0.097575	0.002309	0.002017	0.001047	0.000204	0.003648	0.000970
9000	0.293776	0.135291	0.123749	0.002629	0.002303	0.001200	0.000219	0.004075	0.001075
10000	0.369551	0.170833	0.153605	0.002934	0.002589	0.001396	0.000222	0.004554	0.001182
11000	0.446222	0.212774	0.188153	0.003209	0.002879	0.001405	0.000232	0.004962	0.001292
12000	0.521856	0.270450	0.220040	0.003512	0.003162	0.001614	0.000243	0.005410	0.001430
13000	0.616159	0.322227	0.258422	0.003818	0.003452	0.001700	0.000249	0.005866	0.001546
14000	0.709187	0.396926	0.301389	0.004152	0.003787	0.001883	0.000260	0.006337	0.003543
15000	0.814552	0.456789	0.344512	0.004433	0.004044	0.002086	0.000268	0.006773	0.001782
16000	0.927685	0.525819	0.392671	0.004770	0.004346	0.002107	0.000306	0.007206	0.001884
17000	1.061198	0.614698	0.444797	0.005068	0.004757	0.002397	0.000289	0.007683	0.002015
18000	1.174716	0.700365	0.495186	0.005362	0.004960	0.002682	0.000298	0.008126	0.002117
19000	1.316916	0.826117	0.562290	0.006113	0.005293	0.002590	0.000307	0.008584	0.002199
20000	1.447126	0.930459	0.615576	0.006460	0.005608	0.002770	0.000323	0.009023	0.002266

[[SORTED]]

n	Bubble	Selection	Insertion	Merge	Heap	Quick	Counting	Radix	Bucket
1000	0.001345	0.001454	0.000005	0.000277	0.000213	0.000108	0.000132	0.000457	0.000102
2000	0.005388	0.006406	0.000010	0.000602	0.000499	0.000250	0.000163	0.000965	0.000220
3000	0.013364	0.013411	0.000014	0.000844	0.000808	0.000391	0.000174	0.001350	0.000325
4000	0.023293	0.023311	0.000018	0.001186	0.001093	0.000496	0.000164	0.001803	0.000432
5000	0.035485	0.036292	0.000022	0.001503	0.001422	0.000624	0.000176	0.002251	0.000541
6000	0.050384	0.052519	0.000027	0.001802	0.001721	0.000776	0.000181	0.002728	0.000654
7000	0.067441	0.070788	0.000031	0.002034	0.001870	0.000851	0.000217	0.003156	0.000791
8000	0.087809	0.092524	0.000035	0.002500	0.002295	0.001023	0.000200	0.003623	0.000870
9000	0.110678	0.116929	0.000055	0.002639	0.002597	0.001232	0.000245	0.004120	0.000976
10000	0.138092	0.144299	0.000044	0.003057	0.002839	0.001368	0.000222	0.004507	0.001090
11000	0.171398	0.175519	0.000049	0.003220	0.003108	0.001499	0.000231	0.005013	0.001212
12000	0.195160	0.211179	0.000053	0.003618	0.003516	0.001609	0.000240	0.005413	0.001306
13000	0.229089	0.243472	0.000058	0.004102	0.003820	0.001895	0.000260	0.005861	0.001420
14000	0.265261	0.291513	0.000062	0.004409	0.004052	0.001899	0.000260	0.006312	0.001540
15000	0.313358	0.325703	0.000067	0.004676	0.004437	0.001935	0.000268	0.006807	0.001644
16000	0.346174	0.369335	0.000071	0.005007	0.004692	0.002304	0.000279	0.007212	0.001750
17000	0.389511	0.425895	0.000075	0.005463	0.005047	0.002380	0.000288	0.007696	0.001872
18000	0.437242	0.467527	0.000079	0.005763	0.005406	0.002473	0.000298	0.008107	0.001961
19000	0.495902	0.520328	0.000084	0.006185	0.005673	0.002672	0.000306	0.008773	0.002063
20000	0.538628	0.576809	0.000088	0.006457	0.006266	0.002915	0.000317	0.009154	0.002151

[[NEARLY SORTED]]

n	Bubble	Selection	Insertion	Merge	Heap	Quick	Counting	Radix	Bucket
1000	0.001940	0.001461	0.000248	0.000285	0.000218	0.000118	0.000135	0.000452	0.000104
2000	0.008560	0.005947	0.001086	0.000612	0.000471	0.000241	0.000221	0.000902	0.000207
3000	0.019771	0.013136	0.002575	0.000912	0.000744	0.000366	0.000178	0.001352	0.000351
4000	0.034026	0.023601	0.004150	0.001170	0.001045	0.000522	0.000164	0.001817	0.000495
5000	0.052731	0.042221	0.006203	0.001492	0.001406	0.000677	0.000173	0.002254	0.000580
6000	0.078343	0.052149	0.009107	0.001861	0.001734	0.000796	0.000187	0.002727	0.000707
7000	0.104534	0.070724	0.012258	0.002184	0.002019	0.000969	0.000195	0.003155	0.000870
8000	0.141947	0.092422	0.016387	0.002573	0.002327	0.001115	0.000204	0.003608	0.000933
9000	0.172356	0.116709	0.020540	0.002921	0.002523	0.001216	0.000212	0.004051	0.001018
10000	0.213027	0.150074	0.024072	0.003141	0.002906	0.001413	0.000221	0.004627	0.001188
11000	0.270490	0.175029	0.030821	0.003536	0.003218	0.001606	0.000233	0.004995	0.001273
12000	0.318898	0.208588	0.037093	0.003902	0.003477	0.001778	0.000242	0.005423	0.001408
13000	0.366762	0.244299	0.043180	0.004206	0.003842	0.001810	0.000252	0.005904	0.001652
14000	0.427123	0.285913	0.050250	0.004551	0.004142	0.001901	0.000262	0.006349	0.001605
15000	0.485413	0.331238	0.053628	0.005000	0.004492	0.002231	0.000271	0.006767	0.001721
16000	0.561410	0.368996	0.065337	0.005264	0.004767	0.002377	0.000278	0.007211	0.001860
17000	0.624720	0.416939	0.070894	0.005557	0.005128	0.002591	0.000290	0.007685	0.001942
18000	0.718182	0.468572	0.083607	0.005947	0.005403	0.002649	0.000301	0.008167	0.002191
19000	0.791299	0.522076	0.091370	0.006297	0.005769	0.002750	0.000315	0.008573	0.002164
20000	0.873745	0.577351	0.102622	0.006605	0.006121	0.003360	0.000322	0.009308	0.002291



3 Entrega

Instruções para entrega do seu trabalho:

1. **O que entregar?**

   Um arquivo compactado a ser entregue deve conter o seguinte:

   - programa(s) desenvolvido(s) (e um arquivo **`Makefile`**, se for o caso),
   - tabelas dos tempos de execução dos algoritmos (em texto), de acordo com a formatação apresentada na seção 2.1, e
   - gráficos (no formato **`.pdf`**) gerados a partir das tabelas dos tempos de execução dos algoritmos. Exemplos relativos às tabelas da seção 2.1 sobre o experimento "vetor aleatório", usando o software **`gnuplot`**, são mostrados na Figura 1.

   Compacte todos esses arquivos com o compactador de sua preferência e entregue um único arquivo (com extensão **`.tgz`**, **`.bz2`**, **`.zip`**, **`.rar`**, ...).

2. **Forma de entrega**

   A entrega será realizada diretamente no Sistema (AVA/UFMS), na disciplina de Implementação Algorítmica – T01. Um fórum de discussão deste trabalho já se encontra aberto. Após abrir uma sessão digitando seu *login* e sua senha, vá até a sessão "Atividades" e escolha "Entrega da atividade 2". Você pode entregar a atividade quantas vezes quiser até às **23 horas e 59 minutos** do dia **26 de setembro de 2022**. A última versão entregue é aquela que será corrigida. Encerrado o prazo, não serão mais aceitos trabalhos.

3. **Linguagem de programação**

   Use a linguagem de programação de sua preferência para desenvolver esta atividade.

4. **Atrasos**

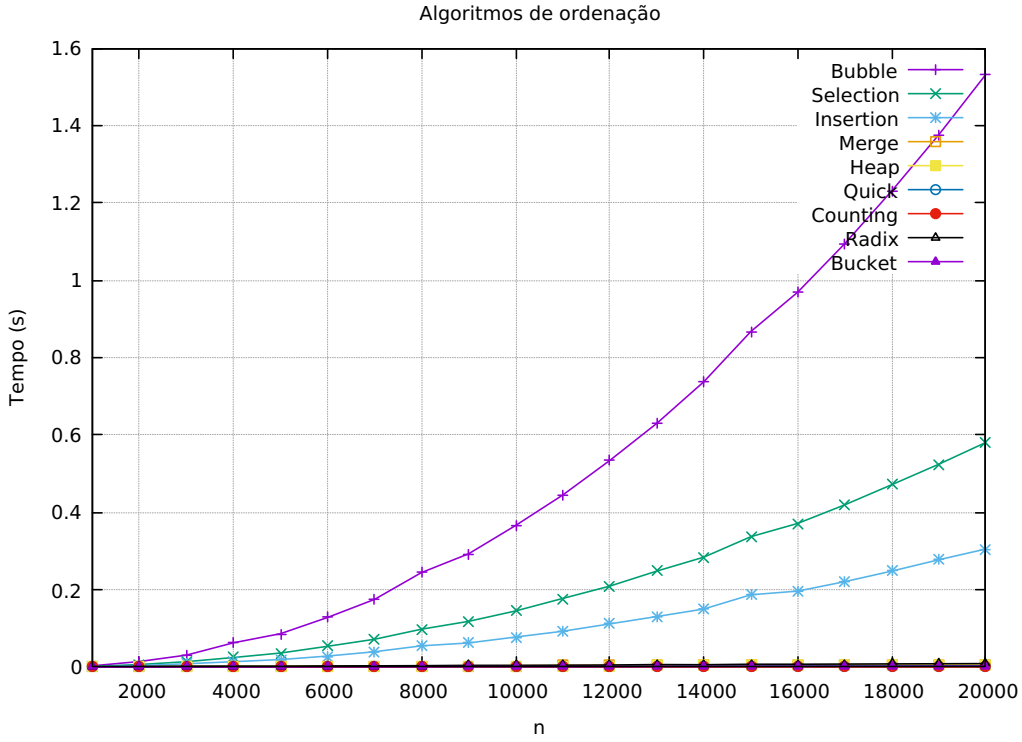
   Trabalhos atrasados não serão aceitos. Não deixe para entregar seu trabalho na última hora. Para prevenir imprevistos como queda de energia, problemas com o sistema, e/ou falha de conexão com a internet, sugerimos que a entrega do trabalho seja feita pelo menos um dia antes do prazo determinado.

5. **Erros**

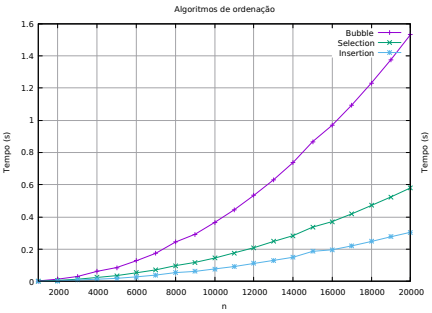
   Trabalhos com erros de compilação/interpretação receberão nota ZERO. Faça todos os testes necessários para garantir que seu programa está livre de erros de compilação/interpretação.

6. **Arquivo com o programa fonte**

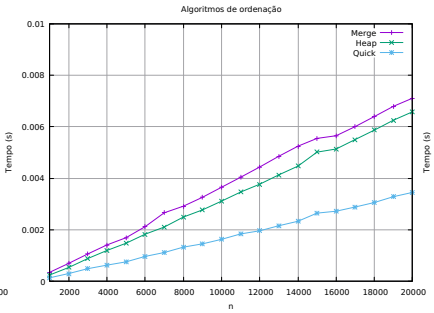
   Seu(s) arquivo(s) contendo o(s) fonte(s) do(s) programa(s) na linguagem escolhida deve(m) estar bem organizado(s). Um programa tem de ser muito bem compreendido por uma pessoa. Verifique se seu programa tem a indentação adequada, se não tem linhas muito longas, se tem variáveis com nomes significativos, entre outros. Não esqueça que um programa bem descrito e bem organizado é a chave de seu sucesso. Também não esqueça da documentação de seu programa.



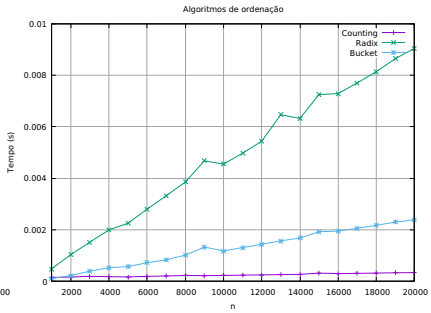
(a) Execução de todos os algoritmos para os casos de teste do experimento "vetor aleatório".



(b) Algoritmos elementares.

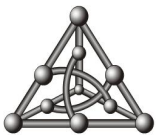


(c) Algoritmos eficientes.



(d) Algoritmos de tempo linear.

Figura 1: Execução dos algoritmos para o experimento "vetor aleatório". Observe que na figura 1a, as linhas dos tempos de execução dos algoritmos eficientes e de tempo linear ficam sobrepostas e não conseguimos distingui-las. Por isso, três outros graficos são mostrados: figuras 1b, 1c e 1d, apresentando os desempenhos dos algoritmos neste experimento divididos em algoritmos elementares, algoritmos mais eficientes (ordenação por comparação) e algoritmos lineares, respectivamente.



7. **Conduta Ética**

   O trabalho deve ser feito INDIVIDUALMENTE/COM SEU GRUPO. Cada estudante tem responsabilidade sobre cópias de seu trabalho, mesmo que parciais. Não compartilhe seu programa ou trechos de seu programa. Você pode consultar seus colegas para esclarecer dúvidas e discutir idéias sobre o trabalho, ao vivo ou no fórum de discussão da disciplina, mas NÃO copie o programa!

   Trabalhos considerados plagiados terão nota ZERO.