UNIVERSIDADE FEDERAL DE MINAS GERAIS

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Algoritmos e Estruturas de Dados III
Trabalho Prático 3 – NuCache

Orlando Enrico Liz Silvério Silva

Belo Horizonte
3 de dezembro de 2017

1 Introdução

Em Ciência da Computação, a localidade de referência, também conhecida como princípio da localidade, é um termo para um fenômeno no qual os mesmos valores, ou locais no armazenamento, são frequentemente acessados, dependendo do padrão de acesso à memória. Há dois tipos de localidade de referência: temporal e espacial. Na primeira, se um item é referenciado, ele tende a ser referenciado novamente. Já na segunda, se um item é referenciado, itens próximos tendem a ser também.

Esse conteúdo pode ser utilizado para melhorar a eficiência de algoritmos, como a multiplicação de matrizes, objeto de interesse do trabalho.

2 Solução do Problema

Esta seção tem como objetivo apresentar as soluções implementadas. Para tanto, inicialmente será introduzida a modelagem proposta, ilustrando as etapas da resolução do problema. Em seguida, serão apresentados detalhes relacionados à localidade de referência.

2.1 Modelagem do Problema

A entrada consiste, na primeira linha, de dois inteiros positivos: N ($1 \leq N \leq 2000$) indicando o número de linhas e colunas das matrizes A, B a serem multiplicadas (que devem ser quadradas) e *bsize* ($1 \leq N \leq 50$), indicando o tamanho do bloco a ser utilizado. Em seguida, haverão N linhas, cada uma con N elementos inteiros contendo os elementos da matriz A. E, por fim, N linhas com N elementos inteiros contendo os elementos da matriz B.

Recebendo as duas matrizes $A \in \mathbb{Z}^{NxN}$ e $B \in \mathbb{Z}^{NxN}$ a serem multiplicadas, o resultado deve ser armazenado em uma matriz $C \in \mathbb{Z}^{NxN}$. O algoritmo, portanto, consiste em realizar o produto interno das linhas de A pelas colunas de B de modo que $C_{ij} = \sum_{k=1}^{N}(A_{i,k}B_{k,j})$.

$$\begin{bmatrix} -1 & 3 \\ 4 & 2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} (-1)\times1+3\times3 & (-1)\times2+3\times4 \\ 4\times1+2\times3 & 4\times2+2\times4 \end{bmatrix} = \begin{bmatrix} 8 & 10 \\ 10 & 16 \end{bmatrix}$$

No *algoritmo*1 essa multiplicação é feita similarmente ao exemplo anterior. Ou seja, em uma matriz NxN são N leituras por elemento e N valores somados por destino.

Os arranjos em C são alocados em linhas. Por isso, andar por colunas em uma linha consiste em acesso a elementos sucessivos (explora localidade espacial). Por outro lado, andar por linhas em uma coluna consiste em acessar elementos distantes e, por isso, a taxa de falhas é alta. Sendo assim, supondo uma cache com blocos de tamanho 32 *bytes* que comporte 4 *double* por bloco, ao executar o algoritmo sem uso de *blocking* em uma matriz 4×4, pode-se dizer que A terá $0,25$ misses por iteração enquanto B terá 1 miss por iteração.

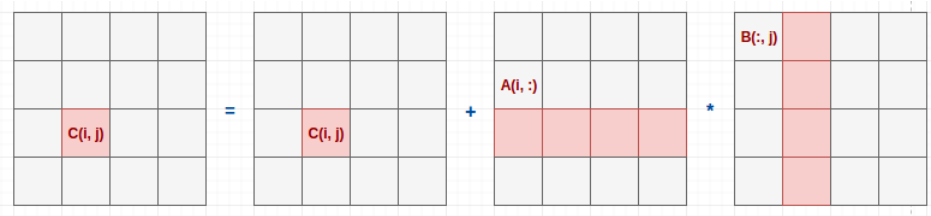

Figura 1 – *Estrutura do Algoritmo de Multiplicação de Matrizes sem blocos.*

Já no *algoritmo*2, essa operação utiliza o conceito de *blocking*, que consiste em organizar as estrututras de dados abstratamente em grandes pedaços (blocos). Desse modo, quando necessário, um bloco é carregado no cache L1. As operações necessárias são realizadas e o bloco é descartado. Carrega-se o próximo e assim por diante.

2.2 Blocos

Algorithm Multiplicação de Matrizes com Blocos

```
for kk = 0; kk < N; kk+ = bsize do
    for jj = 0; jj < N; jj+ = bsize do
        for i = 0; i < N; i + + do
            for j = jj; j < jj + bsize; j + + do
                sum ← C[i][j]
                for k = kk; k < kk + bsize; k + + do
                    sum ← sum + A[i][k] ∗ B[k][j]
                end for
                C[i][j] ← sum
            end for
        end for
    end for
end for
```

A ideia do algoritmo utilizando o conceito de *blocking*, portanto, é definir um tamanho de bloco ($bsize$) a ser carregado na cache, particionar as linhas das matrizes A e C em pedaços de tamanho $1 \times bsize$ e particionar B em blocos de tamanho $bsize \times bsize$. O loop interno (j, k) multiplica o pedaço de A por um bloco de B e acumula o resultado no respectivo pedaço de C. O loop (i) itera sobre as n linhas de A e C usando o mesmo bloco de B. Ou seja, a estratégia explora o fato de que uma matriz pode ser particionada em submatrizes e estas submatrizes podem ser manipuladas como escalares.

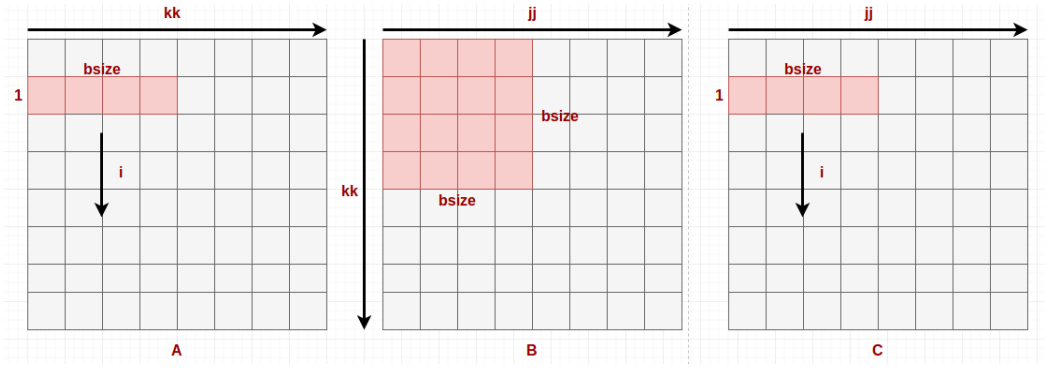

Figura 2 – *Estrutura do Algoritmo de Multiplicação de Matrizes com blocos.*

Se, por exemplo, A, B e C forem matrizes 4×4, pode-se particionar cada matriz em quatro submatrizes de tamanho 2×2 e efetuar a multiplicação.

2.3 Análise de Localidade

Supondo que o tamanho do bloco seja divisor do número de linhas da matriz, que variáveis locais são armazenadas em registradores de modo que não seja necessário instrução de leitura ou escrita para acessar estas variáveis, que a matriz sempre cabe na memória secundária (RAM) e que o bloco sempre cabe na primária (cache) é possível fazer uma análise teórica da localidade temporal e espacial. O tamanho da cache é suficiente para armazenar os blocos que estão sendo multiplicados, ou seja, $|cache| = bsize^2 + 2 * bsize$.

Portanto, as referências à matriz A possuem boa localidade espacial uma vez que cada segmento é acessado com um passo de 1. Além disso, possuem também uma boa localidade temporal já que o segmento inteiro é referenciado $bsize$ vezes em sucessão.

Já as referências a matriz B, por sua vez, possuem boa localidade temporal pois o bloco de tamanho $bsize \times bsize$ é acessado n vezes sucessivamente.

Finalmente, as referências a C têm boa localidade espacial porque cada elemento do segmento é escrito seguidamente. Porém, não possuem uma boa localidade temporal uma vez que cada segmento é acessado apenas uma vez.

2.4 Organização do Código

A organização do código, em termos de arquivos, ficou do seguinte modo:

- **matriz.h e matriz.c**: fornece uma *struct Matriz* que possui uma matriz e uma variável que armazena seu tamanho, assim como funções de alocação e liberação de memória.
- **noBlock.c**: realiza a multiplicação das matrizes A e B sem blocos e armazena em C.
- **block.c**: realiza a multiplicação das matrizes A e B com blocos e armazena em C.

3 Análise de Complexidade

Nesta seção serão apresentadas as análises de complexidade de tempo e espaço dos algoritmos implementados.

3.1 Complexidade Temporal

3.1.1 Algoritmo sem uso de Blocos

Tanto a alocação quanto a liberação de cada matriz ocorre em $O(n^2)$. Já a multiplicação das matrizes A e B é feita em $O(N^3)$.

3.1.2 Algoritmo com Blocos

Tanto a alocação quanto a liberação de cada matriz ocorre em $O(n^2)$. Já para a multiplicação há dois *loops* externos que variam os blocos dentro de B de tamanho *bsize x bsize*. Em seguida um loop i que itera sobre as N linhas de A e C, um loop j para cada coluna do bloco de B (ou seja, *bsize* vezes) e um *loop k* para cada elemento do segmento de A e linha de B (bsize vezes). Juntando tudo, temos:

$$\frac{N}{bsize} \times \frac{N}{bsize} \times N \times bsize \times bsize = N^3$$

Portanto, a complexidade da multiplicação é $O(N^3)$.

3.2 Complexidade Espacial

Em ambos os algoritmos são utilizadas 3 matrizes quadradas de tamanho $N \times N$. Portanto a complexidade é $O(n^2)$ nos dois casos.

4 Análise Experimental

Para analisar o comportamento dos algoritmos desenvolvidos, foi implementado um gerador de testes para diferentes tamanhos de N (200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000) com variações do tamanho de bloco *bsize* (10, 20, 25, 40, 50 - devem dividir os valores de N) para cada valor de N. No caso dessa análise, foram 50 entradas geradas.

4.1 Cache-Miss

O algoritmo de multiplicação de matrizes implementado com o uso de blocos foi submetido a testes de *cache-miss*. Cada entrada de tamanho de matriz NxN e tamanho de bloco *bsize* foi executada 30 vezes para se obter um valor médio do *cache-miss* como estimativa do valor real. Ele foi obtido excutando o comando *perf stat -e cache-misses -o [nomeArquivo] ./block <[nomeTeste]* no terminal do Linux. Todos os resultados obtidos estão disponíveis na seção Apêndice ao final do relatório.

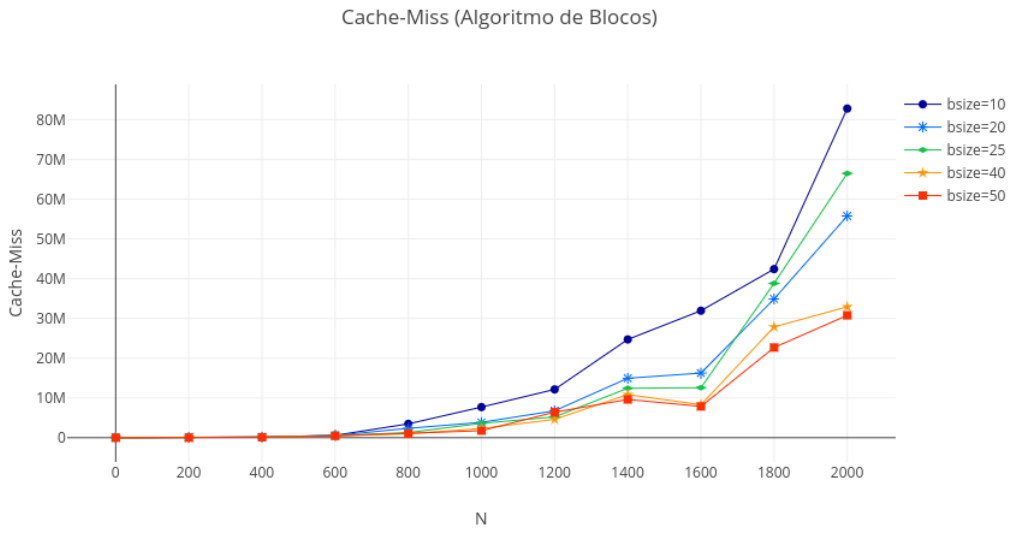

Figura 3 – *Cache-Miss* de acordo com N e *bsize*

É possível notar pelo gráfico gerado que, conforme N se torna maior, para um mesmo tamanho de bloco, ocorrem mais $cache - miss$.

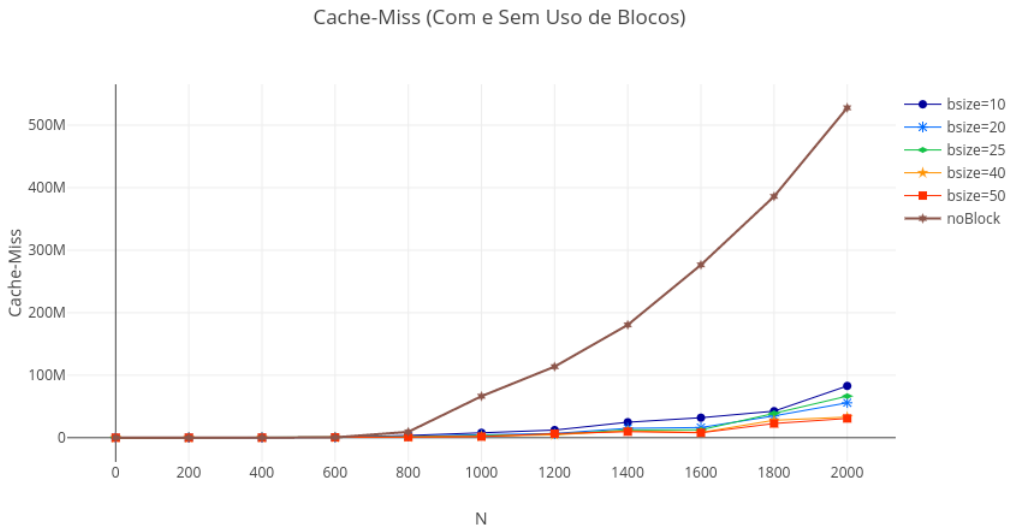

Figura 4 – *Cache-Miss* de ambas as implementações da multiplicação de matrizes

Já para a análise de *cache-miss* no algoritmo sem o uso de *blocking*, como não faz diferença o valor de *bsize*, um único arquivo para cada N foi executado 30 vezes. O gráfico apenas realça a eficiência da técnica de blocos, reduzindo o *cache-miss* drasticamente em relação aos resultados do naive com a melhoria da localidade de referência.

4.2 Tempo

Uma análise do tempo de execução de cada algoritmo com N e *bsize* variando também foi feita de maneira análoga à apresentada na seção anterior. Ou seja, cada arquivo foi executado 30 vezes para se obter uma aproximação do valor real. Os *logs* desses experimentos também estão disponíveis ao final do relatório. O comando */usr/bin/time -f "%e "./Block <[nomeTeste]* foi utilizado para obter o tempo.

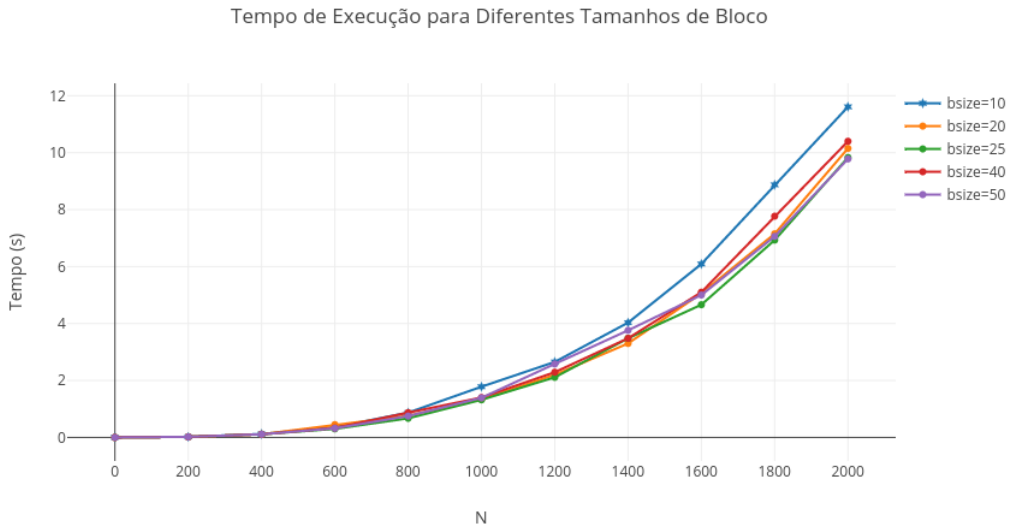

Figura 5 – Tempos de execução com N e *bsize* variando

A Figura 6, apesar de alguns resultados (devidos principalmente a impossibilidade de se obter um valor preciso), mostra uma tendência de blocos maiores levarem menos tempo para executar.

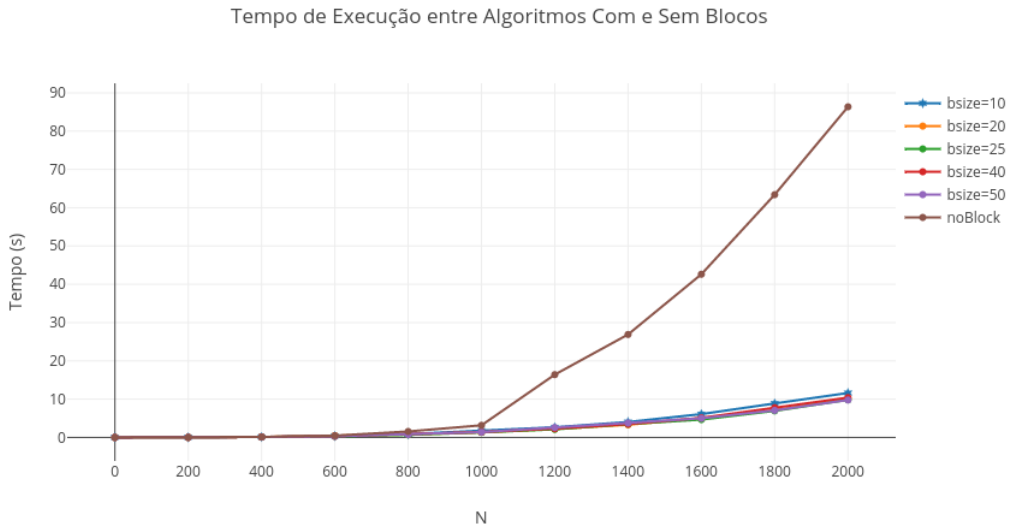

Figura 6 – Tempos de execução entre os algoritmos

Similarmente à comparação de *cache-miss*, o algoritmo que não utiliza blocos para calcular a multiplicação de matrizes possui um resultado consideravelmente inferior em relação ao algoritmo de *blocking*. Apesar disso, para valores de N pequenos, o desempenho de ambos é próximo. O gráfico ilustra o comportamento assintótico do algoritmo e o resultado condiz com as análises de complexidade avaliadas anteriormente, pois as curvas são bem próximas de n^3.

5 Conclusão

Há muitas maneiras de se implementar uma multiplicação de matrizes variando a maneira de acessar a memória. Cada uma tem suas especificidades e sua escolha depende dos objetivos do projeto.

Esse trabalho apresentou duas implementações para a operação em questão. O algoritmo que utiliza a técnica de *blocking* visa diminuir a taxa de *cache-miss* melhorando a localidade temporal e espacial dos acessos à memória.

Um ponto importante é verificado ao constatar que, apesar da implementação mais simples possuir mesma complexidade temporal ($O(n^3)$), ela ainda é consideravelmente mais lenta que o algoritmo que utiliza a técnica de *blocking* para N grande, justamente por não explorar bem o conceito de localidade de referência.

6 Referências Bibliográficas

Cormen, Thomas H. Introduction to Algorithms. Cambridge, Mass: MIT Press, 2001.

ZIVIANI, Nivio. Projeto de Algoritmos com implementações em PASCAL e C, 3 ed. Cengage Learning, 2011.

A Theory Temporal and Spatial Locality. Computer Science Department Illinois University. Acesso em: 30 nov 2017. Disponível em: <http://snir.cs.illinois.edu/PDF/Temporal%20and%20Spatial%20Locality.pdf>.

Study Tonight. Acesso em: 02 nov 2017. Disponível em: <http://www.studytonight.com/computer-architecture/memory-organization>

7 Apêndice

7.1 Cache-Miss do Algoritmo de Blocos

N=200
bsize=10 17433 28951 23270 16827 12354 13831 10441 17198 11618 15936 19577 10982 15195 14142 12369 26446 11100 11243 18203 18699 9373 15791 16604 18271 18220 21839 21289 24770 23173 26552
bsize=20 18646 12888 11366 14071 13687 32896 11586 20961 11308 6804 26877 22577 25466 25343 31667 35945 28068 14388 13700 13106 10534 17649 10020 10258 12531 29619 26092 25711 39470 24580
bsize=25 19201 12019 12956 24799 6897 17368 15411 12451 6241 15586 16333 9146 16262 19095 15306 11986 12989 9507 22828 13895 12125 18445 8362 14951 16679 13537 14451 11721 7833 22488
bsize=40 14526 16362 15235 12411 15937 8034 25278 12165 12420 15836 12813 22106 13339 24069 18157 15369 14467 17925 15079 11321 12802 11922 14652 9045 15591 15479 30506 20068 18302 15878
bsize=50 27275 13235 14248 11025 15081 14166 12801 11764 17509 13729 13819 11020 23163 16308 15332 11451 18669 13170 10640 11774 21130 23773 34378 14125 11568 15068 12488 10324 9931 7987
N=400
bsize=10 160393 136828 200885 102959 155220 234060 234571 114352 132168 163088 164123 134022 113741 111851 125126 192555 210586 177414 111386 103890 134617 130164 123636 200488 214135 217149 169119 178113 148061 122804
bsize=20 144629 211507 139392 256279 200937 311942 147341 107674 149197 227447 153889 214453 131768 218664 206132 218605 228626 196915 207053 370523 83855 149294 159532 119208 121820 135868 189811 144093 104145 101429
bsize=25 122101 80474 81862 150367 108458 151473 139605 142677 139349 124302 116337 93781 165090 157135 196154 97274 104656 119598 177109 102563 105826 159974 149113 95436 174627 149621 176282 124333 77509 141051
bsize=40 132661 145826 165729 131144 158347 132249 91644 162639 167963 148540 131110 130811 134106 135954 133700 149684 163052 154148 128219 120127 151906 156920 169418 131917 63839 134856 176880 188937 207771 132027
bsize=50 120212 198535 136639 46883 104329 88170 133692 81260 74746 89471 116687 131029 79824 99341 62731 71599 108740 87249 138068 128948 106371 83291 79107 109225 153127 107416 85955 80446 113342 106418
N=600
bsize=10 515847 633889 755012 540507 1420411 425490 450550 352644 547955 1256140 541775 466608 601093 454214 468098 775816 431871 548399 425038 539944 525129 670907 564803 466886 532262 794435 453125 474257 495960 420599
bsize=20 470360 487585 400906 405999 425825 385338 448121 427683 368960 384207 463427 417425 389817 739964 330885 1223825 644905 788331 439664 406684 490343 802497 531347 637333 655839 895115 1015491 670044 482147 373050
bsize=25 411369 474957 450420 389418 598979 890846 584048 480454 441646 674827 791796 415661 382648 475527 480907 435900 505890 673053 633936 468306 557514 461508 511847 437473 613006 414170 646732 368217 461173 496828
bsize=40 524094 576849 629177 556148 427501 478129 415164 410392 415494 351959 384331 412707 418533 507978 566648 499691 507599 454943 507761 593241 539671 417767 428104 435996 509476 484815 522843 473725 394255 455067
bsize=50 409412 351415 495894 380693 421063 422257 467213 405300 503308 410915 677594 340049 495178 383606 404203 532559 430799 362800 590270 412254 396129 448375 401098 458135 403052 1104699 403398 400495 420470 447543
N=800
bsize=10 2789804 4284724 2585038 2560997 2651234 4638440 2718640 2843339 2593910 2497565 2713704 3677843 2642226 3540000 2542187 2561369 2636331 4599449 4100740 3714941 4913230 4110138 4672861 4109517 3251197 3139334 4108260 4397726 5363896 3105516
bsize=20 2690084 3316434 1784252 2186575 1793791 2629759 2989550 1648680 2055230 2738781 3822307 1829956 3883099 5878418 1685025 2389346 2559372 2756011 1723360 1938749 1639122

1358774 1410858 2686793 2626769 2869321 1860095 1189579 1354812 1197092
bsize=25 1160623 1170652 1228788 1458183 1494524 1140823 1070036 1105425 1110048 1238444 1589349 1390896 1458843 1390814 1173736 1444767 1155673 1329547 1190023 1191741 1213006 1326225 1294100 1149825 1156640 1256002 1836252 1422253 1344112 1121223
bsize=40 897064 931027 887985 864256 874649 1844763 919749 878293 972944 921782 954415 887563 839716 918369 880138 874210 1271295 869375 974261 874500 885540 853102 892975 1042839 916815 894428 949614 987776 836766 946153
bsize=50 956727 883400 863931 794191 761460 868054 841173 819001 817737 837008 771949 900749 930293 1247436 889945 870520 882889 811781 880979 919345 1078117 2243309 894220 872543 1005982 1312908 1642607 1017462 845302 2683187
N=1000
bsize=10 7737442 6840329 6901220 6785941 6588818 12418991 6441394 6560653 11530904 6439931 6334310 6499801 6482999 6842781 11933169 6446383 6490864 6429986 6530117 11440632 6483133 7406688 6271755 11923207 6365250 7331278 6316621 12151199 6580147 6407438
bsize=20 3350752 3432287 3464666 3391385 3419445 4915065 3475655 3374326 5029787 3622640 3437778 3505579 3538274 5366741 3376452 5298281 5029107 3641841 3732652 3442941 3363891 3532158 3443226 3523343 3521594 3434935 3787995 3422007 3410311 6357564
bsize=25 4017115 2864106 2834218 4045350 2817967 4200611 2871958 2926229 3277915 2865668 3878638 4037379 3855323 4068665 2872055 2802247 3868047 3951275 4104812 3997230 4178482 2850774 2783411 5785297 3490233 2851913 4145121 2807907 4059621 4746667
bsize=40 1957504 2070066 2011240 2565236 2007452 1933144 1992416 2493742 2607891 2489397 1998068 2083019 1980381 1963308 1994535 2651673 1980459 2060942 2652139 2872749 2620634 3192603 2788796 2643862 1980459 2605575 2529965 2045615 2045789 2704166
bsize=50 1740563 2791286 1761561 1772595 1645462 1640794 1620140 1596188 1600252 1610132 1664074 1792745 1841227 2472025 1642870 1646812 2015900 2377505 1587340 1615821 1770776 1659826 1616831 1638176 1626401 1594645 1602769 1604467 1635761 1665288
N=1200
bsize=10 12106842 11973831 12069412 12067335 12069742 11995278 12047658 11985671 12267418 12196051 12169827 12086575 12112005 12186489 12092524 12113085 12037396 12187430 12223712 12168091 12176884 11979038 12508524 12369522 12166753 12135265 12072463 12223744 12092030 12234353
bsize=20 6175175 6192594 6227978 6216972 6157081 6152124 6262550 10613942 7620507 6195722 6871175 6280217 6193268 6177004 6778777 6129209 6325720 6226708 6138009 6234919 6183501 6340921 6236640 6154328 11053039 6148024 6645301 6244206 6231250 10530963
bsize=25 5082521 5038368 5732614 7973769 5008195 5174838 5042682 5332138 5027740 4985726 5035253 5066653 5063641 5316048 5040803 5095750 5064747 5080587 5078871 5048394 5025125 5126060 4985340 5021008 5142370 5116861 5068530 5702539 5090935 5068933
bsize=40 5099017 3330026 3456582 3417684 3322172 3427927 3458338 3398483 3456384 3834941 10008802 7168310 3492330 6419355 3543592 3433437 5572308 4542731 4319146 5045473 4440251 4353064 4489513 5800755 4394741 3683112 4927641 4111247 3519944 8808013
bsize=50 3739151 7494027 8529935 10751300 8694803 5485393 8697701 5172761 4987231 5129610 8786079 7287657 8200571 5187536 5433464 4349180 5488540 4897222 6235238 6128301 6775142 5996573 5200642 8610009 7821970 8715296 5072315 3680484 4829351 5062063
N=1400
bsize=10 23898370 32331929 24309987 23154966 32308144 23050197 22807430 22675080 22323678 22472054 23751060 27012202 21675183 28632958 23484459 22779328 22017533 22228945 23733152 23143245 22780778 23698246 22474752 24473954 23182519 29084395 27222518 23924519 23458783 33096699
bsize=20 14467802 15309997 14852827 21278355 14954307 14376585 13638062 14264146 15198547 14210112 13651814 14549939 14340899 25834826 16246829 13511352 13974182 14461123 13629907 18667182 18997915 13546326 13047327 17204021 10817974 12071927 10923679 13733885 16697668 10037169
bsize=25 8964270 8020901 8324639 11274000 15536515 10720861 16960779 12927460 14951530 9417389 8155270 9529684 9225125 8853972 16834024 22879136 11034430 8041409 7864846 21243251 16863879 10934543 9436931 8094075 13920491 8308029 14953792 22185267 10333133 17134860
bsize=40 6086332 15995234 12552867 6490876 14859565 12746101 12034706 10758641 18095800 9161832 9796436 10494347 9401482 9407400 9184700 9472283 14487271 10324104 8785518 11316951

9139022 8549728 9059440 9037406 8805310 8993746 13958601 17827805 9092018 9427190
bsize=50 9303416 10741381 8821636 8344441 11315075 8638830 9007744 8325873 9132755 10326638
11541688 6325843 5221077 9867484 10946514 13299633 8568252 8935433 8301442 9156230 9168972
16406994 9704501 9666459 9470537 8785824 9733829 9000612 8229199 12121216
N=1600
bsize=10 34507983 35036703 33984984 34016549 34012443 35587224 67906905 29831088 29474230
29533273 29810812 30091807 29563723 29812864 29508941 29149480 30262832 29724526 29748668
29455067 29568302 30006985 29582331 29623865 29341158 29424577 29511441 29654235 30008012
30172142
bsize=20 15111766 15475139 14907151 14999990 14887069 14864849 16171999 15085036 16000797
15310881 15102869 15064699 14822118 14977177 16143647 14805083 16423719 19246671 21122002
18820774 22623930 17937010 17317692 15271367 15290853 18029995 14887189 14985790 16248839
15364337
bsize=25 12549986 12337696 12315194 14163126 12214888 11916443 12063124 12396978 12243479
12599539 12200916 12182497 12228078 13185900 12939600 14604086 12181773 14077740 12245191
11937837 11893496 12346619 12434084 12242712 12294653 12129096 12978284 12406108 13646526
12003746
bsize=40 7994289 9223929 10083061 7780034 8971723 8050741 7763231 7863393 7934674 9114777
7919379 8008768 8249574 7899534 8151187 7937616 9195444 7943074 8691307 8112686 7938548
7870720 7953065 9995935 7921470 7866290 7976887 8057686 7793094 8056602
bsize=50 8397423 7790953 6948649 7040689 6562895 7476930 6730845 7610861 7098489 7013432
6848334 24001830 7797162 6714062 6649206 8081186 6662520 6584465 6749517 6500789 6710340
6505095 8828262 11191599 8565580 6837812 6681770 8092667 6437509 6818191
N=1800
bsize=10 41935736 42510973 42071076 41799396 42813788 43266495 41849291 43542784 41852537
42753770 41864439 42233059 41911440 42230867 41975605 43278308 43260909 42533081 42182346
42104757 42001060 42009106 43163127 42810593 42323446 42485343 41655816 42337119 42523072
42836687
bsize=20 20776480 22110765 20843631 21732822 29190307 33234672 35805050 41182345 38937688
61342487 30670300 29772159 39926514 39180698 26530468 24158317 32075463 37568630 33579491
27047343 35846302 46443570 35716613 39407967 37883246 26589960 39528676 51191764 43821889
44788352
bsize=25 36714578 38057329 46788609 34537672 35770967 30549235 40200086 47894045 57042855
43904660 45340526 37370468 28968678 45339831 46092697 21694844 59482345 33925472 39039727
31032055 31773576 32354381 32954934 34944071 31442716 59959670 33111627 44468508 32405608
31329874
bsize=40 40371884 33744153 23418422 22798924 24438457 19107021 21312934 33981540 26882506
23709507 29461533 30395006 34253612 25933883 26456243 28959678 26832223 28384259 33210274
31182938 31598806 26813641 25802690 31215270 32881512 25783963 20278668 24643619 17020637
35051748
bsize=50 26647148 29864306 23350148 23238888 23498399 23390629 20607018 19137091 26651497
30241888 28536646 31494728 22590086 17708758 16781758 21675676 19049367 19706412 23482404
21038450 23690664 22236680 19484020 18872321 16037409 21406016 19058508 20727610 25542954
24461986
N=2000
bsize=10 67568954 73894266 70271318 63995171 75983060 70539602 84455684 80353804 97596420
92228447 86671397 71194821 69142794 72877781 116144097 92581078 105015587 87934939 73381411
76831730 72111905 142406496 67797793 85452720 95216608 72904212 84907986 79751586 76390702
77660357
bsize=20 54244382 45192811 88393008 52501684 32902975 34055746 59357199 48060451 42668598
43643641 43130307 32216765 55718114 48807912 36595469 47375625 42426739 50019924 59061071
40612525 62893665 119068450 75991666 66617253 52563018 41512334 37891984 34757437 71445337
153058687
bsize=25 86234067 45980993 41250991 66664494 76263221 55918224 42017602 86802235 60124069
73861392 92637582 117389579 72588408 72727277 46114206 64296197 77389416 62443341 97507959
91355216 63926816 97343537 45182658 45774614 43180780 39990730 80587329 49715488 41479917
57035548

```
bsize=40 45941986 43677541 40189347 50104853 34417947 45604238 33213386 48829878 35282093
17072474 19143276 17615510 16557562 17131452 30772028 27166545 41186536 36273047 30662640
38076332 35856528 32398826 33996778 31226192 31075768 33590250 32282294 30516430 29325433
28329559
bsize=50 30418519 28811728 23671267 27867977 28026588 36166508 38941674 28135949 33595952
31415554 36938164 38320510 38260292 37953959 34958861 23348770 23469332 27510369 21781173
18055331 23076426 24429788 38787118 33329786 31418807 26376752 25634446 33920006 36646140
41828896
```

7.2 Tempo de Execução do Algoritmo de Blocos

N=200
bsize=10 0.02 0.02 0.02 0.02 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.02 0.01 0.02 0.02 0.02 0.02 0.04 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.03 0.02 0.02 0.02 0.02
bsize=20 0.02 0.02 0.02 0.01 0.02 0.02 0.02 0.01 0.02 0.02 0.01 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.01 0.02 0.02
bsize=25 0.02 0.02 0.01 0.02 0.02 0.01 0.02 0.02 0.02 0.03 0.02 0.02 0.02 0.01 0.02 0.01 0.02 0.02 0.02 0.01 0.02 0.02 0.02 0.02 0.01 0.01 0.02 0.01 0.02 0.02
bsize=40 0.02 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.03 0.02 0.02 0.02 0.02 0.02
bsize=50 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.01 0.02 0.02 0.02 0.01 0.02
N=400
bsize=10 0.11 0.11 0.11 0.11 0.11 0.11 0.14 0.14 0.13 0.12 0.11 0.11 0.11 0.13 0.15 0.12 0.11 0.12 0.11 0.12 0.14 0.11 0.11 0.11 0.11 0.12 0.11 0.11 0.11 0.12
bsize=20 0.11 0.13 0.10 0.10 0.13 0.11 0.11 0.10 0.11 0.10 0.11 0.10 0.10 0.10 0.11 0.11 0.10 0.11 0.11 0.12 0.10 0.11 0.12 0.11 0.10 0.10 0.12 0.10 0.11 0.10
bsize=25 0.11 0.10 0.10 0.10 0.10 0.10 0.10 0.11 0.10 0.11 0.10 0.11 0.10 0.10 0.11 0.10 0.11 0.10 0.12 0.12 0.11 0.10 0.11 0.10 0.11 0.10 0.11 0.10 0.10 0.11
bsize=40 0.14 0.11 0.11 0.11 0.11 0.12 0.11 0.12 0.11 0.11 0.11 0.11 0.11 0.12 0.11 0.12 0.11 0.11 0.11 0.11 0.11 0.11 0.12 0.11 0.12 0.11 0.12 0.11 0.12 0.11
bsize=50 0.17 0.11 0.12 0.11 0.11 0.11 0.12 0.11 0.12 0.11 0.11 0.11 0.11 0.12 0.11 0.11 0.11 0.11 0.11 0.11 0.11 0.11 0.11 0.11 0.12 0.11 0.11 0.11 0.11 0.11
N=600
bsize=10 0.36 0.33 0.33 0.33 0.34 0.35 0.34 0.34 0.33 0.33 0.34 0.34 0.34 0.34 0.34 0.34 0.33 0.34 0.33 0.35 0.34 0.35 0.35 0.33 0.34 0.33 0.33 0.34 0.87 0.36
bsize=20 4.47 0.30 0.31 0.31 0.36 0.31 0.31 0.36 0.35 0.31 0.31 0.31 0.31 0.36 0.31 0.31 0.31 0.30 0.30 0.30 0.30 0.31 0.31 0.31 0.30 0.31 0.31 0.30 0.31 0.31
bsize=25 0.33 0.31 0.30 0.30 0.31 0.31 0.30 0.30 0.30 0.30 0.30 0.31 0.30 0.31 0.30 0.30 0.30 0.30 0.30 0.31 0.30 0.30 0.30 0.30 0.30 0.30 0.30 0.30 0.30 0.31 0.30
bsize=40 0.33 0.33 0.32 0.33 0.32 0.32 0.33 0.33 0.32 0.32 0.32 0.33 0.33 0.33 0.33 0.33 0.34 0.33 0.33 0.35 0.33 0.33 0.34 0.33 0.33 0.33 0.35 0.34 0.35 0.33
bsize=50 0.33 0.32 0.32 0.31 0.31 0.32 0.31 0.32 0.31 0.32 0.32 0.32 0.31 0.33 0.33 0.31 0.32 0.32 0.32 0.32 0.32 0.31 0.31 0.32 0.31 0.32 0.31 0.33 0.32 0.31
N=800
bsize=10 0.82 0.79 0.77 0.79 0.79 0.79 0.79 0.78 0.78 0.79 0.84 1.15 1.06 1.14 0.98 0.90 0.85 0.81 0.84 0.88 0.80 0.80 0.99 1.00 0.83 0.83 0.84 0.93 0.92 0.85
bsize=20 0.98 0.92 0.74 0.68 0.67 0.69 0.69 0.77 0.74 0.69 1.73 0.73 0.71 0.77 0.67 0.71 0.71 0.71 0.68 0.80 0.68 0.69 0.69 0.69 0.70 0.76 0.68 0.68 0.78 0.68
bsize=25 0.69 0.67 0.67 0.66 0.66 0.68 0.69 0.67 0.66 0.67 0.76 0.69 0.69 0.66 0.66 0.66 0.66 0.65 0.67 0.67 0.69 0.68 0.66 0.67 0.68 0.66 0.66 0.67 0.66 0.67
bsize=40 4.86 0.73 0.71 0.74 0.77 0.78 0.73 0.73 0.74 0.72 0.71 0.73 0.72 0.72 0.72 0.73 0.71 0.73 0.72 0.73 0.76 0.73 0.71 0.73 0.73 0.76 0.72 0.72 0.73 0.72
bsize=50 0.72 0.74 0.70 0.70 0.69 0.69 0.70 0.71 0.71 0.70 0.68 0.79 0.73 0.71 0.68 0.72 0.69 0.72 0.75 1.75 0.74 0.74 0.72 0.72 0.70 0.71 0.71 0.71 0.69 0.71
N=1000
bsize=10 1.55 1.54 1.79 1.78 2.00 1.97 2.16 1.97 2.08 2.30 2.07 1.79 1.75 1.58 1.57 1.54 1.79 2.28 2.13 2.08 1.72 1.56 1.55 1.54 1.56 1.56 1.55 1.53 1.55 1.51
bsize=20 1.46 1.29 1.29 1.28 1.30 1.31 1.29 1.52 1.33 1.34 1.31 1.33 1.32 1.39 1.34 1.31 1.34 1.38 1.32 1.43 1.27 1.30 1.32 1.30 1.28 1.29 1.48 1.32 1.37 1.31
bsize=25 1.27 1.25 1.29 2.25 1.26 1.25 1.28 1.28 1.40 1.26 1.37 1.29 1.25 1.25 1.28 1.28 1.26 1.41 1.27 1.27 1.30 1.32 1.32 1.29 1.28 1.28 1.29 1.33 1.28 1.29
bsize=40 1.44 1.38 1.41 1.41 1.41 1.36 1.47 1.55 1.38 1.35 1.35 1.43 1.38 1.42 1.50 1.61 1.37 1.34 1.37 1.36 1.38 1.35 1.38 1.33 1.33 1.39 1.34 1.38 1.42 1.36
bsize=50 1.34 1.34 1.33 1.32 1.34 1.32 1.33 1.36 1.41 1.29 1.28 1.29 1.37 1.33 1.35 1.36 1.39 1.40 1.38

1.52 2.09 1.55 1.37 1.34 1.29 1.32 1.31 1.53 1.35 1.35
N=1200
bsize=10 2.62 2.65 2.58 2.74 2.70 2.62 2.61 2.58 2.66 2.65 2.57 2.54 2.58 2.63 2.63 2.61 2.67 3.20 2.65 2.59 2.64 2.63 2.65 2.61 2.83 2.61 2.58 2.56 2.54 2.71
bsize=20 2.22 2.13 2.16 2.32 2.26 2.25 2.19 2.29 2.24 2.17 2.14 2.15 2.14 2.31 2.26 2.13 2.18 2.24 2.16 2.22 2.14 2.18 2.15 2.17 2.14 2.22 2.27 2.13 2.26 2.31
bsize=25 2.86 2.36 2.05 2.05 2.05 2.05 2.03 2.04 2.05 2.11 2.00 2.05 2.08 2.01 2.00 2.00 2.00 2.11 2.07 2.05 2.09 2.95 2.08 2.00 2.04 2.03 2.03 2.02 2.01 2.01
bsize=40 2.21 2.19 2.33 2.19 2.18 2.18 2.17 2.18 2.18 2.19 2.16 2.19 2.22 2.29 2.25 2.24 2.20 2.19 2.25 2.25 2.37 2.51 2.39 2.15 2.30 2.32 2.43 2.67 2.65 2.75
bsize=50 2.36 2.68 2.26 2.72 3.02 2.62 2.63 2.47 2.95 3.06 2.92 2.67 2.54 2.74 3.00 3.14 2.54 3.10 2.52 2.50 2.23 2.19 2.33 2.61 2.47 2.30 2.35 2.14 2.12 2.31
N=1400
bsize=10 4.37 4.83 4.64 4.75 4.14 3.92 3.97 3.94 3.88 3.95 3.86 3.95 3.98 3.86 3.87 3.88 3.85 3.86 3.86 3.98 4.03 4.11 3.90 3.87 3.89 3.89 3.89 3.93 3.94 4.01
bsize=20 4.67 3.21 3.20 3.44 3.27 3.26 3.28 3.49 3.23 3.22 3.21 3.20 3.19 3.23 3.37 3.20 3.26 3.19 3.37 3.25 3.23 3.21 3.19 3.27 3.23 3.21 3.40 3.18 3.21 3.19
bsize=25 3.25 3.14 3.58 3.13 3.15 3.10 3.13 3.14 3.12 3.20 3.18 3.73 3.11 3.54 4.26 3.71 3.73 3.99 4.21 4.36 4.03 3.71 3.45 3.20 3.27 3.26 3.72 3.72 3.24 3.17
bsize=40 5.89 3.40 3.37 3.40 3.49 3.41 3.44 3.40 3.43 3.36 3.40 3.40 3.51 3.39 3.40 3.35 3.37 3.40 3.44 3.38 3.38 3.42 3.37 3.42 3.38 3.36 3.35 3.39 3.39 3.40
bsize=50 3.37 3.29 3.29 3.43 3.63 3.48 3.98 4.04 3.90 4.80 4.17 3.66 4.03 4.04 3.31 3.64 3.87 3.44 3.88 3.74 4.13 3.61 3.46 3.33 3.96 4.11 3.56 3.71 3.56 4.38
N=1600
bsize=10 8.06 7.78 8.12 7.02 5.95 7.10 5.84 5.84 5.82 6.28 5.82 5.99 5.83 6.05 6.36 5.92 5.63 5.59 5.62 5.69 5.60 5.75 5.60 5.60 5.60 5.64 5.60 5.62 5.82 5.63
bsize=20 4.67 4.98 4.65 4.63 5.76 4.77 4.63 4.60 4.82 4.73 4.99 4.85 4.76 4.77 4.64 5.27 5.30 5.04 6.29 6.50 5.65 5.22 5.40 5.90 5.24 4.89 4.69 4.66 4.72 4.75
bsize=25 5.03 5.25 4.70 4.67 4.82 4.62 4.84 4.63 4.49 4.66 4.63 4.53 4.58 4.64 4.80 4.68 4.85 4.47 4.68 4.51 4.56 4.57 4.56 4.54 4.59 4.56 4.58 4.53 4.64 4.52
bsize=40 5.26 4.87 4.85 4.96 4.88 4.88 4.88 4.88 5.02 4.95 5.23 5.15 5.10 4.98 5.04 5.16 5.10 4.86 4.95 5.14 5.50 5.46 5.25 5.62 5.43 5.04 5.07 4.98 5.34 5.04
bsize=50 5.25 5.03 4.84 5.02 5.40 5.02 4.72 4.74 4.81 4.86 4.80 5.29 5.13 4.91 4.80 4.74 4.81 4.91 5.23 5.09 4.95 5.37 4.77 5.03 4.94 4.90 4.88 5.09 5.10 5.46
N=1800
bsize=10 10.10 9.46 10.06 7.84 8.65 8.18 8.99 9.82 8.62 9.38 9.63 8.36 8.58 8.77 8.90 9.16 9.03 8.87 8.84 8.12 8.46 8.38 9.34 8.98 8.42 8.41 7.94 7.80 8.72 10.20
bsize=20 7.13 6.63 6.86 6.62 6.70 6.60 6.96 6.70 7.00 6.83 7.17 7.63 7.46 7.55 7.26 7.61 8.21 6.62 7.30 6.88 8.59 7.21 7.71 8.10 7.07 6.65 6.94 7.01 6.89 6.74
bsize=25 7.83 8.49 7.19 6.37 6.26 6.57 7.37 6.62 7.06 7.69 7.33 7.68 6.79 7.50 7.56 6.79 7.57 6.93 6.58 7.06 7.32 6.70 6.46 6.26 6.29 6.26 6.26 6.47 6.26 6.27
bsize=40 7.42 7.06 7.35 7.67 7.93 8.02 8.64 7.32 8.52 7.54 8.49 7.89 8.55 8.02 8.13 8.60 8.35 7.47 8.15 8.97 8.37 7.02 7.01 7.23 7.48 7.18 7.29 7.18 7.14 7.23
bsize=50 7.30 6.92 7.21 7.73 7.05 6.90 6.93 7.00 7.11 6.99 6.93 7.03 7.01 7.30 7.16 7.03 6.91 6.90 7.06 6.99 6.91 6.95 6.94 7.14 7.02 6.90 6.99 7.35 7.12 7.09
N=2000
bsize=10 10.77 10.89 10.74 10.75 10.79 10.78 10.74 10.82 10.71 10.73 15.09 13.65 12.41 12.79 11.65 11.72 11.20 12.44 12.98 11.18 11.55 10.79 11.19 11.64 11.90 11.22 11.22 12.47 11.93 11.53
bsize=20 13.26 9.83 9.33 8.92 8.96 10.83 9.47 10.88 10.57 9.04 10.97 10.01 10.50 10.81 9.18 10.95 10.18 9.39 9.52 9.83 10.71 9.87 10.93 9.27 10.53 10.90 9.91 10.70 9.79 9.53
bsize=25 9.23 11.89 9.27 9.76 9.87 9.88 10.26 9.83 10.23 10.39 9.80 9.04 10.23 8.85 9.96 10.00 9.10 8.79 10.62 10.17 10.93 9.48 8.92 9.61 9.63 9.61 10.13 9.46 10.18 9.83
bsize=40 10.11 10.59 10.35 9.58 9.42 9.95 10.83 10.91 9.66 11.18 10.74 9.87 9.78 9.68 10.08 9.83 10.17 9.82 9.98 9.99 9.82 9.70 10.55 10.54 10.52 10.21 9.33 9.37 9.28 9.36
bsize=50 9.66 8.92 8.92 9.01 9.84 9.50 9.40 9.26 9.65 9.65 10.08 10.16 9.95 10.06 10.96 9.74 10.16 9.65 9.32 9.22 9.09 9.85 10.21 9.89 10.56 10.08 10.22 9.95 10.29 10.28

7.3 Cache-Miss do Algoritmo sem Blocos

N=200
23190 20558 20607 17348 20048 19372 20499 28187 19465 19934 18111 20575 22350 25856 21799
19138 18389 17246 19531 19181 16679 13910 17180 14854 15261 14109 15452 15115 13427 13338
N=400
126225 95332 110459 123432 158040 172386 123888 182164 139798 163239 147072 91176 91469
101468 252156 136024 118736 91445 87990 117243 100082 85613 86821 90837 93890 103965 91555
123298 82254 94034
N=600
309564 332585 207647 301547 253309 1988483 316757 945671 293679 392419 440815 235824 232821
304181 571817 558095 516407 334830 207610 216957 361955 248525 287324 525141 261400 335385
237132 332441 256962 233434
N=800
9187494 8216744 14610182 9190574 8356215 8189499 8301056 8306623 7776470 7830081 9468918
7476278 7861136 7940435 6538528 14123250 8705456 9857668 8227865 7768331 9999594 11256376
8741783 9084255 10279537 9480610 7286379 12254784 13220968 8244238
N=1000
50412686 52223018 50546309 51409293 81500813 85548014 91691460 76203697 50979769 89106310
49460841 87148174 82662974 50433677 51803219 52528220 88843793 51226648 66881645 49595231
84330076 83018618 50119997 52294688 79029563 71561952 50204318 78433137 79499492 49311918
N=1200
107204304 107055522 110048436 110765263 108920948 105700412 107440681 149860524 114776075
113376302 115633038 107235554 108621711 107587876 108219135 110408832 109786723 109312325
107375910 108803472 151296593 109875215 106511163 166217004 108130223 107511032 107945782
108350370 107385010 108229029
N=1400
175317838 173206903 175644803 173357722 174588282 174696698 172711133 174789455 174667599
174760025 174967663 174674868 174918173 174293732 173718533 173222261 174802235 250663255
209884160 172780683 176823020 175178476 173109682 242682322 174368041 174429035 173748958
175797471 175031878 175502809
N=1600
272144076 262645463 260376626 266386748 262792937 261661333 376295862 263304196 262232386
262169409 262782593 264719653 264670666 263063265 261815330 262442655 268265120 262624522
411575259 261171064 263084821 260243700 261865722 262223406 261171593 399946770 262759290
263158112 263491297 262591549
N=1800
376393181 376620123 376571903 378614582 376404865 374954633 375723743 376998073 444168794
377659641 375414738 373384534 382644058 375068550 375584933 373091933 373671478 373391111
375352269 375226862 378400185 374993367 376302584 374202818 454125544 536667997 373036242
374973264 376067050 373594745
N=2000
517041428 513569382 517324871 512634804 521249703 517803442 515503363 524969284 514512356
515476652 516246362 517767394 515428524 514421605 513097826 511219024 516996636 516282343
513775929 515779423 516383330 515585889 517509237 514972536 514885359 513254960 515345441
513863165 520300001 870047172

7.4 Tempo de Execução do Algoritmo sem Blocos

N=200
bsize=10 0.02 0.02 0.02 0.04 0.02 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02
N=400
bsize=10 0.14 0.13 0.13 0.13 0.13 0.13 0.13 0.14 0.13 0.13 0.13 0.13 0.13 0.13 0.13 0.13 0.13 0.13 0.13 0.13 0.13 0.13 0.13 0.13 0.13 0.13 0.13 0.14 0.13 0.13
N=600
bsize=10 0.46 0.46 0.46 0.46 0.46 0.47 0.47 0.49 0.52 0.49 0.48 0.46 0.47 0.49 0.50 0.46 0.47 0.47 0.47 0.46 0.50 0.47 0.46 0.47 0.47 0.46 0.46 0.47 0.46 0.49
N=800
bsize=10 1.24 1.40 1.30 1.27 1.32 1.51 1.36 1.71 1.54 1.61 1.63 1.56 1.27 1.81 1.54 2.44 1.44 1.34 1.40 1.63 1.67 1.54 1.28 1.32 1.58 1.39 1.88 1.82 1.87 1.74
N=1000
bsize=10 3.32 3.08 4.13 3.04 4.06 3.81 2.94 2.91 3.19 3.54 3.23 2.90 4.06 2.98 3.02 2.92 2.73 2.82 2.76 2.77 3.02 2.84 2.85 3.14 2.91 2.88 2.97 2.97 2.98 4.01
N=1200
bsize=10 16.36 16.05 16.11 16.44 16.35 16.15 15.91 16.18 16.39 16.21 16.16 16.65 15.92 16.61 16.45 16.15 16.21 16.09 16.57 16.64 16.48 17.97 17.11 16.72 15.80 16.17 16.18 16.20 17.38 16.31
N=1400
bsize=10 27.57 26.60 26.48 26.49 27.16 26.60 26.50 26.63 26.80 26.63 26.49 27.66 27.20 26.37 26.32 26.38 26.37 28.42 27.63 26.80 26.49 26.59 26.61 26.40 26.64 26.55 26.48 27.01 27.37 29.43
N=1600
bsize=10 39.99 39.69 39.60 40.88 40.12 39.73 44.08 42.87 43.35 39.76 43.65 42.36 44.34 42.49 44.05 44.82 42.97 42.74 43.09 43.26 43.45 42.85 43.74 45.29 45.25 42.08 44.69 44.29 41.67 41.53
N=1800
bsize=10 63.99 61.47 65.35 60.25 62.45 62.03 62.69 62.22 61.90 63.37 63.94 64.75 64.24 65.26 62.49 61.65 61.37 63.66 64.12 65.74 62.51 60.87 60.52 57.57 64.68 63.67 69.18 68.68 69.37 62.38
N=2000
bsize=10 90.31 92.19 93.22 89.57 88.33 88.50 88.33 88.72 90.66 82.88 84.26 84.10 82.01 84.35 83.62 86.45 84.29 85.31 90.77 84.84 82.61 83.62 80.56 82.03 82.32 83.99 87.83 91.72 84.45 89.01