Licenciatura em Engenharia Informática

Algoritmos e Estruturas de Dados

word_ladder

João Catarino
NMec: 93096

Rúben Garrido
NMec: 107927

Nuno Vieira
NMec: 107283

7 de janeiro de 2023

Índice

1 Introdução

Texto aqui

2 Código

2.1 Função hash_table_grow que testa o melhor incremento

```c
static void hash_table_grow(hash_table_t *hash_table)
{
  unsigned int    i;
  double        j;
  unsigned int    k;
  unsigned int    test_new_size;
  unsigned int    test_new_key;
  hash_table_node_t *next;
  hash_table_node_t *node;
  hash_table_node_t **test_new_table;
  unsigned int    colnum;
  unsigned int    free_entries;

  // Determine size_inc based on collision count
  if (hash_table->number_of_collisions > 0 && (hash_table->
   hash_table_size / hash_table->number_of_collisions) < 5)
  {
    // Find the best j
    printf("\nFinding best j. Current hash_table_size is %u.\n",
   hash_table->hash_table_size);
    printf("  j   | new size | memory | free m | colnum\n");
    for (j = 1.1; j < 3; j += 0.005)
    {
      colnum = 0u;
      free_entries = 0u;
      test_new_size = (double)hash_table->hash_table_size * j;
      test_new_table = (hash_table_node_t **)calloc(test_new_size,
   sizeof(hash_table_node_t *));

      for (i=0; i < hash_table->hash_table_size; i++)
      {
        for (node = hash_table->heads[i]; node; node = next)
        {
          test_new_key = crc32(node->word) % test_new_size;
          next = node->next;
          if (test_new_table[test_new_key])
          {
            colnum++;
          }
          test_new_table[test_new_key] = node;
        }
      }
      for (k=0; k < test_new_size; k++) {
        if (!test_new_table[k]) {
          free_entries++;
        }
      }
      printf("%3.3f | %8u | %6lu | %6lu | %6u\n", j, test_new_size,
   test_new_size * sizeof(hash_table_node_t *), free_entries * sizeof(
   hash_table_node_t *), colnum);
    }
  }
}
```

2.2 Script MATLAB que gera os gráficos para análise da hash_table_grow

```
% Get data from file
table = load("first.txt");
j = table(:,1);
new_size = table(:,2);
memory = table(:,3);
free_memory = table(:,4);
collisions = table(:,5);

% Sort free_memory & collisions arrays, based on free_memory
[free_memory_sorted,sortIdx] = sort(free_memory,'ascend');
collisions_sorted = collisions(sortIdx);

% Get ratios
ratio_col_mem = collisions./memory;
ratio_col_free = collisions./free_memory;

% Plots
figure(1)
plot(memory,collisions)
xlabel('Total memory (bytes)')
ylabel('Collisions')
grid on

figure(2)
plot(free_memory_sorted,collisions_sorted)
xlabel('Free memory (bytes)')
ylabel('Collisions')
grid on
xlim([5000 20000])

figure(3)
plot(j,ratio_col_mem)
xlabel('Increment, j')
ylabel('Collisions/Total memory ratio')
grid on

figure(4)
plot(j,ratio_col_free)
xlabel('Increment, j')
ylabel('Collisions/Free memory ratio')
grid on

figure(5)
plot(memory,free_memory)
xlabel('Total memory (bytes)')
ylabel('Free memory (bytes)')
grid on
```