과제명	Written Homework 3
담당교수	배성일
분반	2
학번	B617065
이름	신성우

1. Solution

For this homework, use the data file words.dat. We want to investigate questions regarding the design of a good hash table for the data.

1. Consider the following hash function hash():

```
int hash(char key[5], int hash_prime)
{
  int       i;
  long long x;

  x = 0;
  for (i = 0; i < 4; ++i)
  {
    x = x + key[i];
    x = x << 8;
  }
  x = x + key[4];
  return (x % hash_prime);
}
```

When we put the 5757 five-letter words into a hash table, how many collisions occur for the following values for HASH PRIME?

$$M1 = 7000,\ M3 = 12000,\ M5 = 22000,$$

$$M2 = 6997,\ M4 = 11117,\ M6 = 22307.$$

Note that M1, M3, M5 are composite numbers and M2, M4, M6 are prime numbers. For this question, you probably need to write a computer program. For a hash prime M, the hash function h has a value between 0 and M 1. Let ci be the number of words w, among 5757 words in words.dat, such that h(w) = i. Then the number of collisions is defined to be

$$\sum_{i=0}^{M-1} max\{(c_i - 1), 0\}.$$

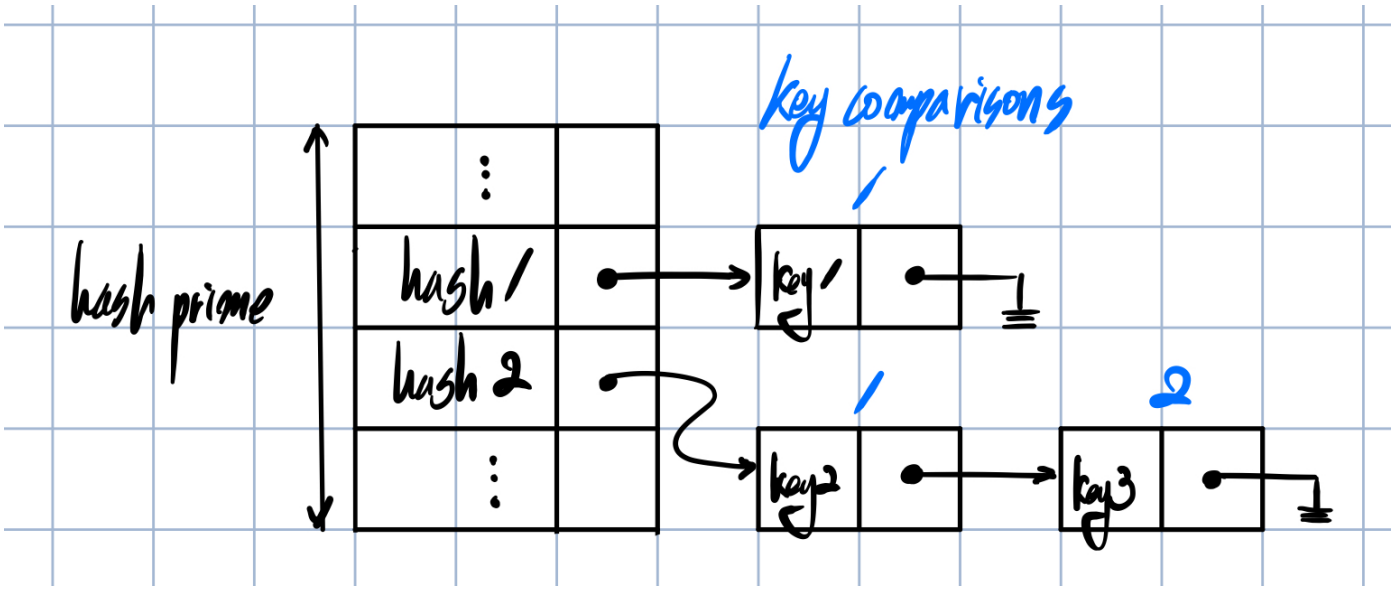
Experiment result

```
For all following experiments, the number of words(keys): 5757
For HASH_PRIME: 7000(composite), then the number of collisions: 2305
For HASH_PRIME: 6997(prime), then the number of collisions: 1821
For HASH_PRIME: 12000(composite), then the number of collisions: 2733
For HASH_PRIME: 11117(prime), then the number of collisions: 1235
For HASH_PRIME: 22000(composite), then the number of collisions: 1359
For HASH_PRIME: 22307(prime), then the number of collisions: 655
For HASH_PRIME: 11117, then the number of key comparisons: 7216
```

위의 실험결과를 통해 얻을 수 있는 결론은 HASH_PRIME의 값이 클수록 대체로 the number of collisions 값이 작아진다는 것이다. 다만, HASH_PRIME의 값을 소수로 설정하는 경우 (HASH PRIME 값을 그와 비슷한 소수가 아닌 수로 설정하는 경우와 비교하여) the number of collisions 값은 대폭 감소한다. 특히 HASH PRIME을 11117로 설정하는 경우와 22000로 설정하는 경우를 비교하면 전자의 경우가 HASH PRIME의 값은 작지만, the number of collisions 값은 오히려 작음을 관찰 할 수 있다. 따라서 HASH PRIME의 값을 적절한 소수로 지정하는 것은 hash collision을 방지하는데 효과적이라고 할 수 있다.

---

2. Supose that we use “chaining” for our hash table, using hash() and HASH PRIME = M4. After all the words are inserted in the hash table, if we search each and every word in words.dat “exactly once”, how many key comparisons are made in total?



위 그림을 참고하여, After all the words are inserted in the hash table, search each and every word “exactly once"인 경우, key comparisions 값은 다음과 같이 정리할 수 있다.

$$\sum_{i=0}^{(HASH\ PRIME)-1} \sum_{j=0}^{c_i} j$$

나아가 위의 공식을 적용한 실험결과를 통해 본 문제의 답은 7216임을 확인할 수 있다.

2. Source code

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void  read_file(char words[5757][5])
{
  FILE  *fp;
  int   i;
  char  buf[5];
  char  dummy[100];

  fp = fopen("words.dat", "r");
  if (fp == NULL)
    exit(1);
  i = 0;
  while (i < 4)
  {
    fgets(dummy, 100, fp);
    ++i;
  }
  i = 0;
  while (i < 5757)
  {
    fgets(words[i], 6, fp);
    fgets(dummy, 100, fp);
    ++i;
  }
  fclose(fp);
}

int get_hash(char key[5], int hash_prime)
{
  int       i;
  long long x;

  x = 0;
  for (i = 0; i < 4; ++i)
  {
    x = x + key[i];
    x = x << 8;
  }
  x = x + key[4];
  return (x % hash_prime);
}
```

```c
size_t  get_sum(int hash_prime, int *mark, int flag_question_nbr)
{
  int     i;
  size_t  sum;

  i = 0;
  sum = 0;
  while (i < hash_prime)
  {
    if (flag_question_nbr == 1)
    {
      if (mark[i] > 1)
        sum += (mark[i] - 1);
    }
    else
      sum += (mark[i] * (mark[i] + 1) / 2);
    ++i;
  }
  return (sum);
}

size_t  process_m(int hash_prime, char words[5757][5], int flag_question_nbr)
{
  int    *mark;
  int     i;
  size_t  sum;

  mark = malloc(sizeof(int) * hash_prime);
  if (mark == NULL)
    exit (1);
  i = 0;
  while (i < hash_prime)
  {
    mark[i] = 0;
    ++i;
  }
  i = 0;
  while (i < 5757)
  {
    ++(mark[get_hash(words[i], hash_prime)]);
    ++i;
  }
  sum = get_sum(hash_prime, mark, flag_question_nbr);
  free(mark);
  mark = NULL;
  return (sum);
}
```

```c
int main(void)
{
  char  words[5757][5];

  read_file(words);
  printf("For all following experiments, the number of words(keys): 5757\n");
  printf("For HASH_PRIME: 7000(composite), then the number of collisions: ");
  printf("%ld\n", process_m(7000, words, 1));
  printf("For HASH_PRIME: 6997(prime), then the number of collisions: ");
  printf("%ld\n", process_m(6997, words, 1));
  printf("For HASH_PRIME: 12000(composite), then the number of collisions: ");
  printf("%ld\n", process_m(12000, words, 1));
  printf("For HASH_PRIME: 11117(prime), then the number of collisions: ");
  printf("%ld\n", process_m(11117, words, 1));
  printf("For HASH_PRIME: 22000(composite), then the number of collisions: ");
  printf("%ld\n", process_m(22000, words, 1));
  printf("For HASH_PRIME: 22307(prime), then the number of collisions: ");
  printf("%ld\n", process_m(22307, words, 1));
  printf("For HASH_PRIME: 11117, then the number of key comparisons: ");
  printf("%ld\n", process_m(11117, words, 2));
  return (0);
}
```

3. How to compile and execute the project

- To compile

```
gcc main.c
```

- To execute

```
./a.out words.dat
```