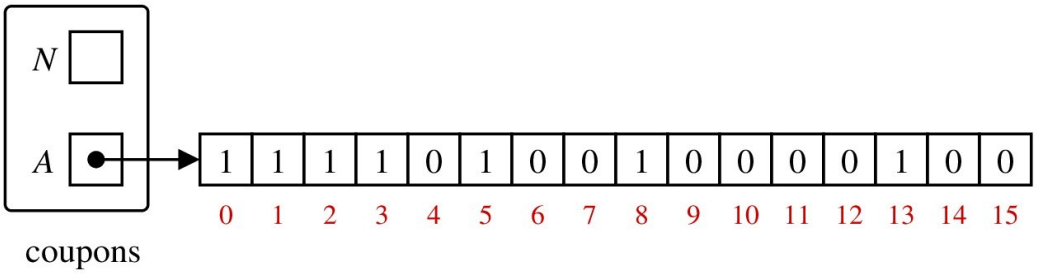
CS29206 Systems Programming Laboratory
Spring 2024
Lab Assignment: 5
Date: 07–Mar–2024

Handling implementation inefficiencies using gprof

This assignment deals with the same problem as Assignment 1. Foobar Chocolate Company (FCC) inserts a randomly chosen coupon from a set of N types of coupons in each chocolate packet. The problem is to simulate a sequence of purchases of chocolate packets until all types of coupons are available to the buyer so that (s)he can demand a gift from FCC. The expected number of purchases for this to happen is NH_N.

A solution is supplied to you in the two files *bitter.c* and *choco.c*. The first file makes an implementation of the set ADT. Here, the coupon types are numbered as 0, 1, 2, … , $N-1$. We maintain an int variable N (the number of coupon types), and a dynamically allocated array A of N int variables storing which types of coupons are in the buyer's collection. Earlier, we have used a single array of $N + 1$ integers to store these items together. Now, we use a structure `coupons` to store N and A separately. The following figure demonstrates the storage of the set {0, 1, 2, 3, 5, 8, 13} with $N = 16$.



The file *bitter.c* defines only the set ADT functions required for solving the given problem. The other file *choco.c* implements the main function, a trial function, and a function to compute harmonic numbers.

A bash script *run* is also supplied. Give the file execute permission, and run as `./run bitter N t` to obtain the gprof output. Here, `N` is the number of coupon types, and `t` is the number of trials. Your code should run for at least 10 seconds for gprof to supply decent statistical results. Adjust `t` accordingly. Note that *bitter.c* is a rather sloppy implementation for solving the given problem. The gprof flat profile will identify the inefficient function(s).

Copy the file *bitter.c* to *sweet.c*. Improve the performance of *bitter* + *choco* in your *sweet* + *choco*. Measure the performance of *sweet* + *choco* as `./run sweet N t`. When should you stop? Well, there is no end to it, perhaps. Researchers may report 0.001% improvements (over the last published results). For you, continue before it is too late to make submissions. Here are the do's and don'ts in your effort.

- You are not allowed to change *choco.c*. This means (among other things) that the I/O behavior of the `coupons` functions cannot be changed.
- You can change the functions in *sweet.c* without changing the given data structure `coupons`.
- You can redesign `coupons` using another suitable data structure and rewrite the functions in *sweet.c* (without changing the function prototypes).
- Assume that N is in the range 10 to 10^5.
- Do not supply `.c` in `run`. The call `./run sweet.c N t` will bite you, albeit not too hard.
- If needed, you may write other helper functions in *sweet.c*.

Submit only your *sweet.c*. We will run it with the *choco.c* supplied to you. Evaluation will depend on how much improvement you achieve and on the algorithm you use in your submitted file.

Sample Output

We have four implementations: bitter, dark, white, and sweet. In all the runs below, we take $N = 10^4$. The number of trials t is adjusted for the different implementations so that the program runs for at least 10 seconds. Note that *choco.c* also computes the average time per trial using the `clock()` function. This is not 100% accurate. Well, gprof too is not 100% accurate, but it will supply you with detailed function-by-function estimates.

```
$ ./run bitter 10000 10
Average number of purchases per trial = 97684.200000
Theoretical average                   = 97876.060360
Time per trial (in microseconds)      = 2025576.700000
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
100.78     20.37    20.37   976852     0.00     0.00  allcoupons
  0.05     20.38     0.01       10     0.00     2.04  nexttrial
  0.00     20.38     0.00   976842     0.00     0.00  addcoupon
  0.00     20.38     0.00       10     0.00     0.00  destroycoupons
  0.00     20.38     0.00       10     0.00     0.00  initempty
  0.00     20.38     0.00        1     0.00     0.00  H
  0.00     20.38     0.00                             __do_global_dtors_aux
  0.00     20.38     0.00                             __gmon_start__
  0.00     20.38     0.00                             __libc_csu_fini
  0.00     20.38     0.00                             __libc_csu_init
  0.00     20.38     0.00                             _dl_relocate_static_pie
  0.00     20.38     0.00                             _fini
  0.00     20.38     0.00                             _init
  0.00     20.38     0.00                             _start
  0.00     20.38     0.00                             atexit
  0.00     20.38     0.00                             data_start
  0.00     20.38     0.00                             deregister_tm_clones
  0.00     20.38     0.00                             etext
  0.00     20.38     0.00                             frame_dummy
  0.00     20.38     0.00                             main
  0.00     20.38     0.00                             register_tm_clones
$ ./run dark 10000 100
Average number of purchases per trial = 99320.020000
Theoretical average                   = 97876.060360
Time per trial (in microseconds)      = 197880.630000
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
100.09     19.51    19.51  9932102     0.00     0.00  allcoupons
  0.57     19.62     0.11      100     1.11   196.54  nexttrial
  0.16     19.65     0.03  9932002     0.00     0.00  addcoupon
  0.03     19.65     0.01      100     0.05     0.05  destroycoupons
  0.00     19.65     0.00      100     0.00     0.00  initempty
  0.00     19.65     0.00        1     0.00     0.00  H
  0.00     19.65     0.00                             __do_global_dtors_aux
  0.00     19.65     0.00                             __gmon_start__
  0.00     19.65     0.00                             __libc_csu_fini
  0.00     19.65     0.00                             __libc_csu_init
  0.00     19.65     0.00                             _dl_relocate_static_pie
  0.00     19.65     0.00                             _fini
  0.00     19.65     0.00                             _init
  0.00     19.65     0.00                             _start
  0.00     19.65     0.00                             atexit
  0.00     19.65     0.00                             data_start
  0.00     19.65     0.00                             deregister_tm_clones
  0.00     19.65     0.00                             etext
  0.00     19.65     0.00                             frame_dummy
  0.00     19.65     0.00                             main
  0.00     19.65     0.00                             register_tm_clones
$ /run white 10000 1000
Average number of purchases per trial = 98073.475000
Theoretical average                   = 97876.060360
Time per trial (in microseconds)      = 14977.923000
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 92.84     11.11    11.11 98073475     0.00     0.00  addcoupon
  6.06     11.84     0.73     1000     0.73    12.06  nexttrial
  1.52     12.02     0.18 98074475     0.00     0.00  allcoupons
```

```
  0.34     12.06     0.04     1000     0.04     0.04  helper
  0.00     12.06     0.00     1000     0.00     0.04  destroycoupons
  0.00     12.06     0.00     1000     0.00     0.00  initempty
  0.00     12.06     0.00        1     0.00     0.00  H
  0.00     12.06     0.00                             __do_global_dtors_aux
  0.00     12.06     0.00                             __gmon_start__
  0.00     12.06     0.00                             __libc_csu_fini
  0.00     12.06     0.00                             __libc_csu_init
  0.00     12.06     0.00                             _dl_relocate_static_pie
  0.00     12.06     0.00                             _fini
  0.00     12.06     0.00                             _init
  0.00     12.06     0.00                             _start
  0.00     12.06     0.00                             atexit
  0.00     12.06     0.00                             data_start
  0.00     12.06     0.00                             deregister_tm_clones
  0.00     12.06     0.00                             etext
  0.00     12.06     0.00                             frame_dummy
  0.00     12.06     0.00                             main
  0.00     12.06     0.00                             register_tm_clones
$ ./run sweet 10000 2500
Average number of purchases per trial = 98111.468800
Theoretical average                   = 97876.060360
Time per trial (in microseconds)      = 5782.816000
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 79.22      7.02     7.02 245281172     0.00     0.00  allcoupons
  9.89      7.89     0.88     2500     0.35     3.52  nexttrial
  9.09      8.70     0.81 245278672     0.00     0.00  addcoupon
  0.74      8.77     0.07     2500     0.03     0.03  destroycoupons
  0.34      8.80     0.03     2500     0.01     0.01  initempty
  0.00      8.80     0.00        1     0.00     0.00  H
  0.00      8.80     0.00                             __do_global_dtors_aux
  0.00      8.80     0.00                             __gmon_start__
  0.00      8.80     0.00                             __libc_csu_fini
  0.00      8.80     0.00                             __libc_csu_init
  0.00      8.80     0.00                             _dl_relocate_static_pie
  0.00      8.80     0.00                             _fini
  0.00      8.80     0.00                             _init
  0.00      8.80     0.00                             _start
  0.00      8.80     0.00                             atexit
  0.00      8.80     0.00                             data_start
  0.00      8.80     0.00                             deregister_tm_clones
  0.00      8.80     0.00                             etext
  0.00      8.80     0.00                             frame_dummy
  0.00      8.80     0.00                             main
  0.00      8.80     0.00                             register_tm_clones
$
```

Naturally enough, we are going to submit our last version.