第一部分

a. 40MB 的記憶體有點難觀察，所以我 malloc()了大約 3GB 之後開 htop 可以明顯看到系統的動作。（可以調整 N 來決定配多少記憶體）

b. code:

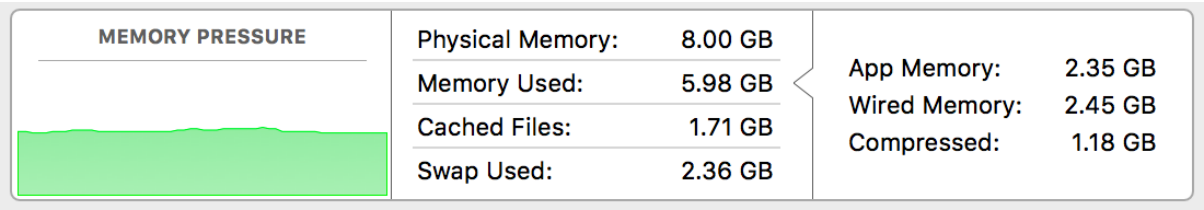
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N (int)1e9  // 這樣 htop 比較好觀察

int main()
{
   srand(time(NULL));
   int n, *data = malloc(sizeof(int) * N); // 大約會拿到 3GB
   scanf("%d", &n); // 輸入數字之前，系統不會配記憶體給程式，看 htop 很明顯

   for (int i = 0; i < N; i++) // 開始用之後，系統就配記憶體了
      data[i] = rand();

   int idx = rand() % N;
   printf("data[%d] = %d\n", idx, data[idx]);

   scanf("%d", &n);

   free(data);

   return 0;
}
```

第二部分

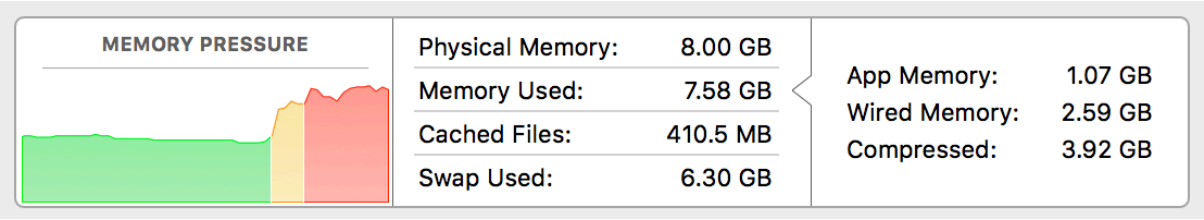
我的做法是我跟電腦 malloc() 20 次的 380 MB，並在每次要完後對裡面進行讀寫。基本上程式運行中，會發現 memory 被吃掉，swap space 用來越多。運行完後，memory 會比程式運行前多出一些記憶體。

實驗結果如下：

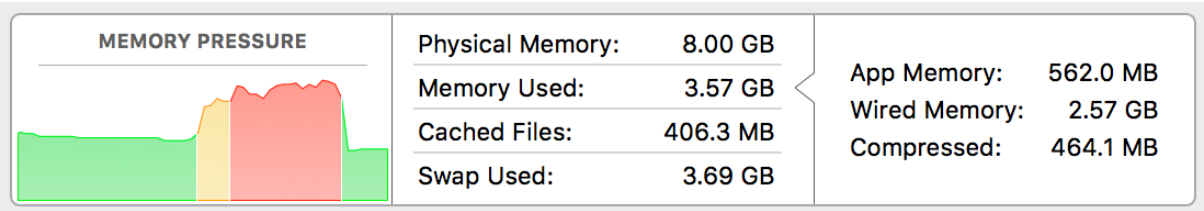
我在自己的電腦上跑，整體執行起來大約花 30 秒鐘。
第一張圖說明原本 OS 正用掉 5.98GB 的 main memory。



第二張圖說明程式運行中會造成 main memory 幾乎被吃光（8GB 用了 7.58GB）



第三張圖說明，程式結束後，main memory 會比原本還多出一些空間來（5.98GB -> 3.57GB，多出了 2.41GB，然後 swap 的使用量有上升一些）。




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

#define N (int)1e8
#define M 20

typedef long long ll;

int main()
{
   srand(time(NULL));

   int *ptr[M];
   for(int i = 0; i < M; i++) {
                        printf("Now on %d\n", i);
      ptr[i] = malloc(N * sizeof(int));
      if (ptr[i] == NULL) {
```

```
            break;
        }

        for(int j = 0; j < N; j++)
            ptr[i][j] = rand() % N;
    }

    for(int i = 0; i < M; i++) {
        free(ptr[i]);
    }

    return 0;
}
```