0.1

C code

```c
//
//  d-way heap.c
//
//
//

#include "dwayheap.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <strings.h>

int size;
int l = 0;

Heap create_heap(int n, int d)
{
    Heap myheap;
    myheap.d = d;
    myheap.size = n;
    myheap.data = (int*)malloc(sizeof(int) * n);

    return myheap;
}



void inserth(Heap *h, int v, int d)
{
    int i = (l + 1);
    int t = h->data[i];
    int pindex = (i-1)/d;
    h->data[i] = v;
    l ++;

    while (h->data[pindex] < h->data[i]){
        int k = h->data[pindex];
        h->data[pindex] = h->data[i];
        h->data[i] = k;
        i = pindex;
        pindex = (i-1)/d;
    }

}

void printHeap(Heap *h){
    printf("\n\nPrinting the heap...\n ");
    printf("\n");
    for (int i=0; i < (h->size); i++){
        printf(" %d ", h->data[i]);
    }
    printf("\n");

}


Heap randominsert(Heap *h, int n, int d)
{
    // empty array
    int* arr = malloc(n * sizeof(int));
```

```
    int k = n-1;

    // initial range of numbers
    for(int i=0;i < n;++i){
        arr[i]=i+1;
    }


    // seed random number generator
    srand(time(NULL));
    int j;

    // mix up numbers
    for (int i = n-1; i >= 0; --i){
        j = rand() % (i +1);
        int m = arr[i];
        arr[i] = arr[j];
        arr[j] = m;
    }

    // output; printing array elements
    for (int i = 0; i <= k; i++) {
        printf("\nheap[%d] = %d", i, arr[i]);
    }

    // first value of the heap
    h->data[0] = arr[0];
    l = 0; // index of this value

    // insert elements one by one into heap
    for (int i = 1; i <= k; i++) {
        inserth(h, arr[i], d);
    }

    return *h;

}


int main()

{
    // we must record the time for a binary heap to be constructed

    int ways;
    printf("\nChoose a value of n: ");
    scanf("%d", &size);

    // get value of d from user; determines the number of 'ways' for heap
    printf("Choose a value of d: ");
    scanf("%d", &ways);

    // create heap
    Heap iheap = create_heap(size, ways);

    Heap fheap = randominsert(&iheap, size, ways);

    printHeap(&fheap);
    return 0;
}
```

```
Choose a value of n: 5
Choose a value of d: 2

heap[0] = 3
heap[1] = 2
heap[2] = 4
heap[3] = 5
heap[4] = 1

Printing the heap...

 5  4  3  2  1

 Time spent: 0.000282 s
```

Figure 1: Sample output for code.

0.2 Results and Discussion

Fig.1 — Average Build Time (3 runs) vs. Values of n with *d=2*

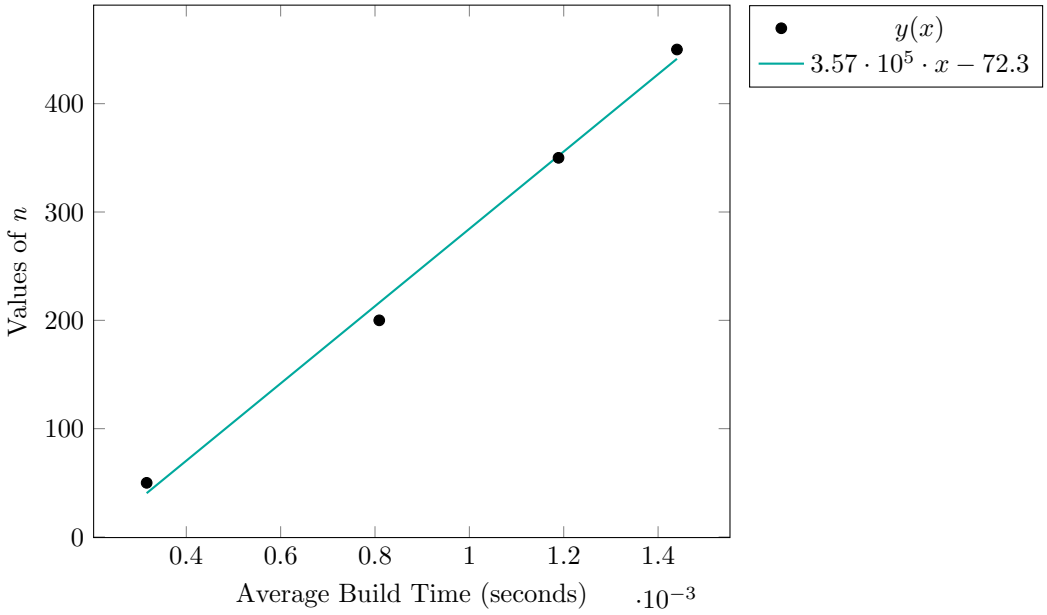

Fig.2 — Average Build Time (3 runs) vs. Values of n with *d=3*

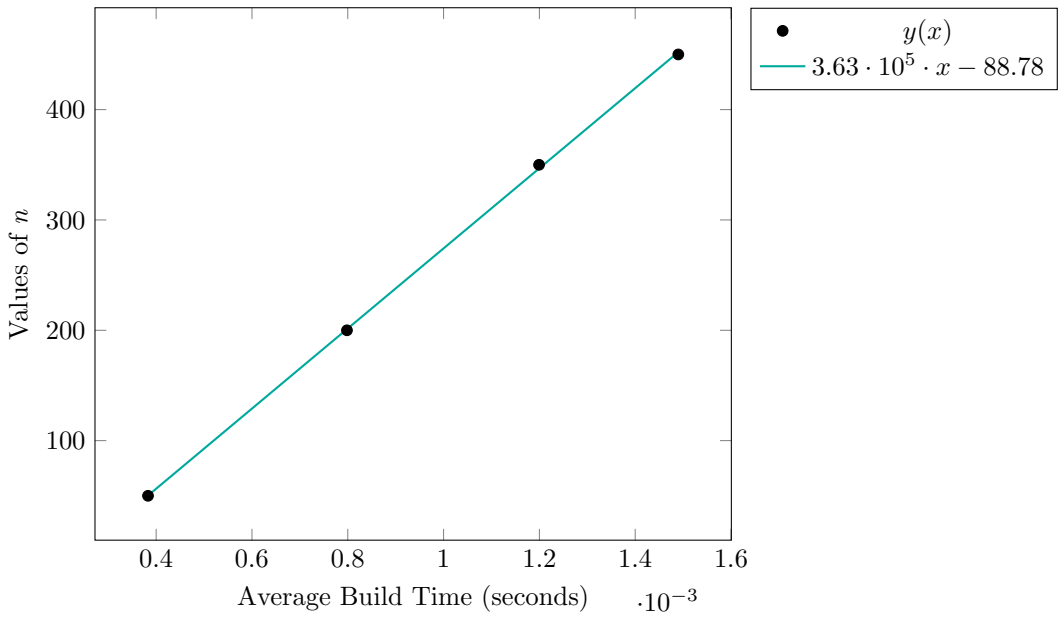

Fig.3 — Average Build Time (3 runs) vs. Values of n with *d=4*

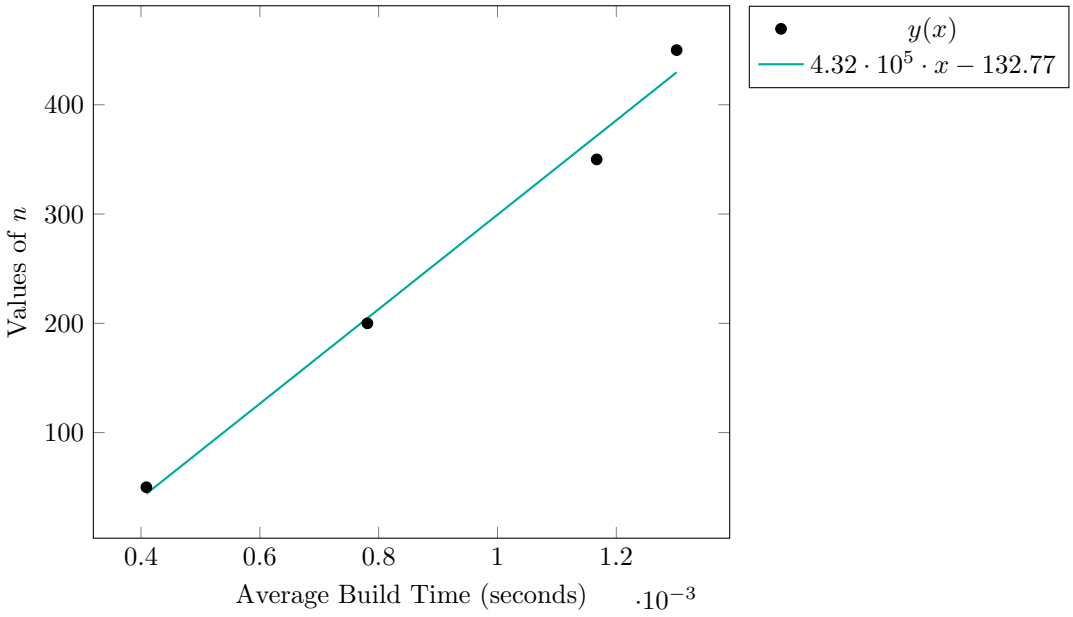

Table 1: Build Time for Varying Values of d and n

Value of n	Value of d	Build Time (s)
50	2	.000316
50	3	.000383
50	4	.000409
200	2	.000809
200	3	.0007983
200	4	.000781
350	2	.001189
350	3	.0011993
350	4	.001167
450	2	.001440
450	3	.001490
450	4	.0013016

Value of d for best performance For these values of d = *2,3,4*, *d=4* gives the best data, although more test runs for each value of d and perhaps also for another value of d such as *d=5* would corroborate these results.