COMPSCI 308: Design and Analysis of Algorithms Homework 1

Luyao Wang

March 25, 2024

1. Sorting

(a) Programming

1. Implement insertion sort as a function

```
void insertionSort(std::vector<int> &arr)
{
  int n = arr.size();

  for (int i = 1; i < n; ++i)
  {
    int key = arr[i];
    int j = i - 1;

    while (j >= 0 && arr[j] > key)
    {
      arr[j + 1] = arr[j];
      j = j - 1;
    }

    arr[j + 1] = key;
  }
}
```

2. Implement a function of a random array with integer values of a given size n

```
std::vector<int> generateRandomArray(int n = 10)
{
  std::random_device rd;
  std::mt19937 gen(rd());
  std::uniform_int_distribution<> dis(-INT_MIN, INT_MAX);

  std::vector<int> result(n);

  std::generate(result.begin(), result.end(), std::bind(dis, gen));

  return result;
}
```

(b) Analysis

Below is the result of the experiment.

Table 1: Running time for different n

n	**milliseconds**
100	0
3545	21
6990	54
10434	100
13879	178
17324	275
20769	402
24214	552
27659	705
31103	893
34548	1096
37993	1321
41438	1580
44883	1850
48328	2151
51772	2520
55217	2877
58662	3209
62107	3596
65552	4093
68997	4695
72441	4875
75886	5365
79331	5897
82776	6451
86221	6990
89666	7549
93110	8312
96555	8777
100000	9413

Figure 1 clearly demonstrates a noticeable trend of rapidly increasing running time as the value of n grows larger.
In Figure 2, I presented a plot depicting the relationship between the square root of the running time, denoted as $\sqrt{T}$, and the input size n. The plot reveals a linear relationship between these two variables. This outcome aligns well with the expected worst-case performance of insertion sort, which is $O(n^2)$.

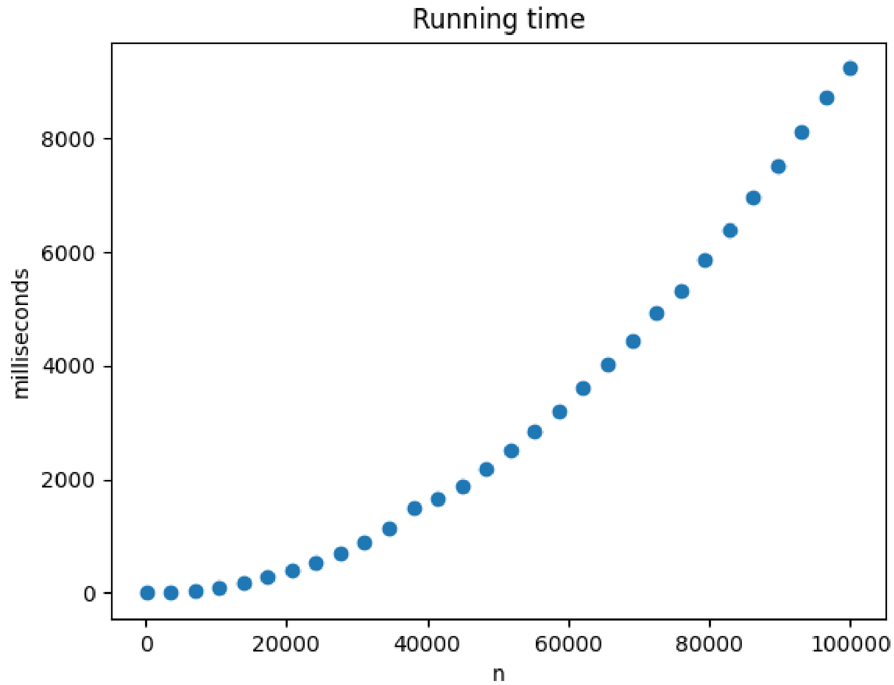


Figure 1: Running time (milliseconds) for different n

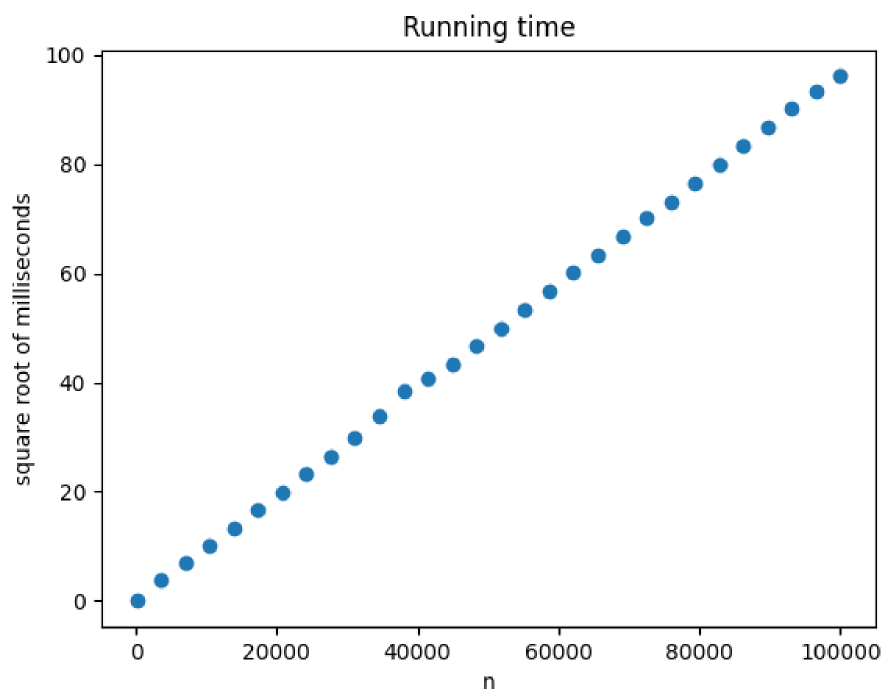


Figure 2: Square root of running time (milliseconds) for different n

2. Asymptotic Notations

We first compare $2^{\sqrt{\log_2 n}}$, $2^{5\log_2 n}$, $\log_2 n$

Substitute $\log_2 n$ with x, since both $\log_2 n$ and x is monotonically increasing and $\lim_{x\to\infty} x = \infty$, $\lim_{n\to\infty} \log_2 n = \infty$

We can compare $\lim_{n\to\infty} \frac{2^{\sqrt{\log_2 n}}}{\log_2 n}$ by $\lim_{x\to\infty} \frac{2^{\sqrt{x}}}{x}$

By L'Hopital's rule, $\lim_{x\to\infty} \frac{2^{\sqrt{x}}}{x} = \lim_{x\to\infty} \frac{\ln 2 \cdot 2^{\sqrt{x}} \cdot \frac{1}{2\sqrt{x}}}{1}$

substitute $\sqrt{x} = t$, we have $\lim_{t\to\infty} \ln 2 \cdot 2^{t} \cdot \frac{1}{2t}$

By L'Hopital's rule again,

we have $\lim_{t\to\infty} \frac{(\ln 2)^2 2^t}{2} = \infty$

① Therefore, $2^{\sqrt{\log_2 n}}$ grows faster than $\log_2 n$

We now compare $2^{\sqrt{\log_2 n}}$ with $2^{5\log_2 n}$ by comparing $2^{\sqrt{x}}$ and 2^{5x}

$\lim_{x\to\infty} \frac{2^{5x}}{2^{\sqrt{x}}} = 2^{5x-\sqrt{x}} = \infty$

② Therefore, $2^{5\log_2 n}$ grows faster than $2^{\sqrt{\log_2 n}}$

$\sqrt{2^n} = (2^n)^{\frac{1}{2}} = 2^{\frac{n}{2}}$

$\lim_{n\to\infty} \frac{\sqrt{2^n}}{2^{5\log_2 n}} = \lim_{n\to\infty} \frac{2^{\frac{n}{2}}}{2^{5\log_2 n}} = \lim_{n\to\infty} 2^{\frac{n}{2} - 5\log_2 n}$

When n is at ∞, $\frac{n}{2} - 5\log_2 n$ is also at ∞, so $2^{\frac{n}{2} - 5\log_2 n}$ is at ∞

③ Therefore, $\sqrt{2^n}$ grows faster than $2^{5\log_2 n}$

$4^{\log_2 n} = 2^n$

$\lim_{n\to\infty} \frac{2^n}{\sqrt{2^n}} = \frac{2^n}{2^{\frac{n}{2}}} = 2^{\frac{n}{2}} = \infty$

④ Therefore, $4^{\log_2 n}$ grows faster than $\sqrt{2^n}$

9999 is constant, so it is smallest in order of growth

In conclusion, order of growth: $4^{\log_2 n} > \sqrt{2^n} > 2^{5\log_2 n} > 2^{\sqrt{\log_2 n}} > \log_2 n > 9999$

3. Proof

(1) Compare 3^n with $c \cdot 2^n$

for $n > \log_{\frac{3}{2}} c$, $3^n > c \cdot 2^n$, contrary to $3^n = O(2^n)$

(2) $\lim_{n\to\infty} \frac{n^2}{n\log^2 n} = \frac{n}{\log^2 n}$

by L'Hopital's rule, $\lim_{n\to\infty} \frac{n}{\log^2 n} = \lim_{n\to\infty} \frac{1}{2\log n \cdot \frac{1}{n}} = \lim_{n\to\infty} \frac{n}{2\log n}$

by L'Hopital's rule, $\lim_{n\to\infty} \frac{n}{2\log n} = \lim_{n\to\infty} \frac{1}{\frac{2}{n}} = \lim_{n\to\infty} \frac{n}{2} = \infty$

Therefore, n^2 grows faster than $n\log^2 n$, contrary to $n^2 = O(n\log^2 n)$

(3) solve: $5n^3 + 120n^2 + 0.99 \le cn^4$

$c \geq \frac{5n^3 + 120n^2 + 0.99}{n^4}$

$c \geq \frac{5}{n} + \frac{120}{n^2} + \frac{0.99}{n^4}$

we find $n_0 = 1$ and $c = 120$ satisfy

Therefore, $5n^3 + 120n^2 + 0.99 = O(n^4)$ holds

(4) solve for $c_1 n^d \le (c+n)^d \le c_2 n^d$

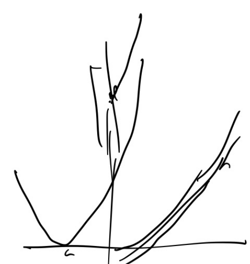
$\ln c_1 n^d \le \ln (c+n)^d \le \ln c_2 n^d$

$\ln (c_1^{\frac{1}{d}} n)^d \le \ln (c+n)^d \le \ln (c_2^{\frac{1}{d}} n)^d$

$d \ln c_1^{\frac{1}{d}} n \le d \ln (c+n) \le d \ln c_2^{\frac{1}{d}} n$

$\ln c_1^{\frac{1}{d}} n \le \ln (c+n) \le \ln c_2^{\frac{1}{d}} n$

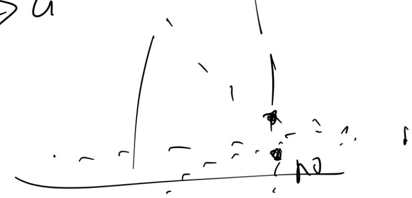
$c_1^{\frac{1}{d}} n \le c+n \le c_2^{\frac{1}{d}} n$



$$\begin{cases} c_1 \le \left(\frac{c+n}{n}\right)^d \\ c_2 \geq \left(\frac{c+n}{n}\right)^d \end{cases} \Rightarrow c_1$$

Consider function $f(x) = \left(\frac{c+x}{x}\right)^d$

$f(x) = \left(1 + \frac{c}{x}\right)^d$



Consider function $g(x) = x^d$. For $x>0$, when $d>0$, $g(x)$ monotonically increase, when $d<0$, $g(x)$ monotonically decrease.

Consider function $t(x) = 1 + \frac{c}{x}$, For $x>0$, when $c>0$, $t(x)$ monotonically decrease, when $c<0$, $t(x)$ monotonically increase

For $x>0$, when $c>0$, $t(x)$ always larger than 0

when $c<0$, $t(x)$ larger than 0 when $x > -c$

when $c>0$, $x>0$, $t(x)$ converges to 1 from top
when $c<0$, $x>0$, $t(x)$ converges to 1 from bottom

① $c>0$, $d>0$, $f(n)$ monotonically decreases and converges to 1 from top

for any $n_0>0$, take $\begin{cases} c_1 \le f(n)_{min} = 1 \\ c_1 \ge f(n)_{max} = \left(\frac{c+n_0}{n_0}\right)^d \end{cases}$

② $c>0$, $d<0$, $f(n)$ monotonically increases and converges to 1 from bottom

for any $n_0>0$, take $\begin{cases} c_1 \le f(n)_{min} = \left(\frac{c+n_0}{n_0}\right)^d \\ c_1 \ge f(n)_{max} = 1 \end{cases}$

③ $c<0$, $d>0$, $f(n)$ monotonically increases and converges to 1 from bottom

for any $n_0>-c$, take $\begin{cases} c_1 \le f(n)_{min} = \left(\frac{c+n_0}{n_0}\right)^d \\ c_1 \ge f(n)_{max} = 1 \end{cases}$

④ $c<0$, $d<0$, $f(n)$ monotonically decreases and converges to 1 from top

for any $n_0>-c$, take $\begin{cases} c_1 \le f(n)_{min} = 1 \\ c_1 \ge f(n)_{max} = \left(\frac{c+n_0}{n_0}\right)^d \end{cases}$

The trivial case when c or d is 0 holds obviously.
In conclusion, $(c+n)^d = \Theta(n^d)$ holds

(5) $\lim_{n\to\infty} \frac{2^{2n}}{2^n} = \lim_{n\to\infty} 2^n = \infty$

2^{2n} grows faster than 2^n, the upper bound relation $2^{2n} = O(2^n)$ cannot hold.

4. Evaluation

For the following code snippet:

```
1 for i to n:
2     j = 1
3     while j < sqrt{n}:
4         j = j + (2 * k)
```

Analysis of execution times for each line:

line number	cost	times
1	c_1	n + 1
2	c_2	n
3	c_3	$\sum_{i=1}^{n} t_i$
4	c_4	$\sum_{i=1}^{n}(t_i - 1)$

where t_i denotes that the number of times the while loop **test** in line 3 is executed for that value of i. t_i is determined by k and n:

$$t_i = \lceil \frac{\sqrt{n} - 1}{2k} \rceil$$

Calculating the worst-case running time:

$$\begin{aligned} T(n) &= c_1(n+1) + c_2 n + c3 \sum_{i=1}^{n} t_i + c_4 \sum_{i=1}^{n} (t_i - 1) \\ &= c_1(n+1) + c_2 n + c3 \sum_{i=1}^{n} \lceil \frac{\sqrt{n} - 1}{2k} \rceil + c_4 \sum_{i=1}^{n} (\lceil \frac{\sqrt{n} - 1}{2k} \rceil - 1) \end{aligned}$$

The highest order is $\frac{n^{1.5}}{k}$.