Algorithms and Data Structures (CSci 115)

California State University Fresno
College of Science and Mathematics
Department of Computer Science
H. Cecotti

Where are we?

- Theoretical complexity of algorithm
  - Asymptotic notation
  - Big O
    - O(1) : execute in the **same** time (or space) regardless of the size of the input data set (block of instruction)
    - O(n) : performance will grow **linearly** and in direct proportion to the size of the input data set (for loop)
    - O(n^2) : performance is directly proportional to the **square** of the size of the input data set (nested loops, 2 for loops)
    - O(log n): divide the problem into sub-problems… more difficult to catch → recursion
      - Example: binary search
  - Recursive functions
    - Master theorem
      - Can you apply it? Yes/No , Which case? 1/2/3 → Asymptotic notation of the function

Learning outcome

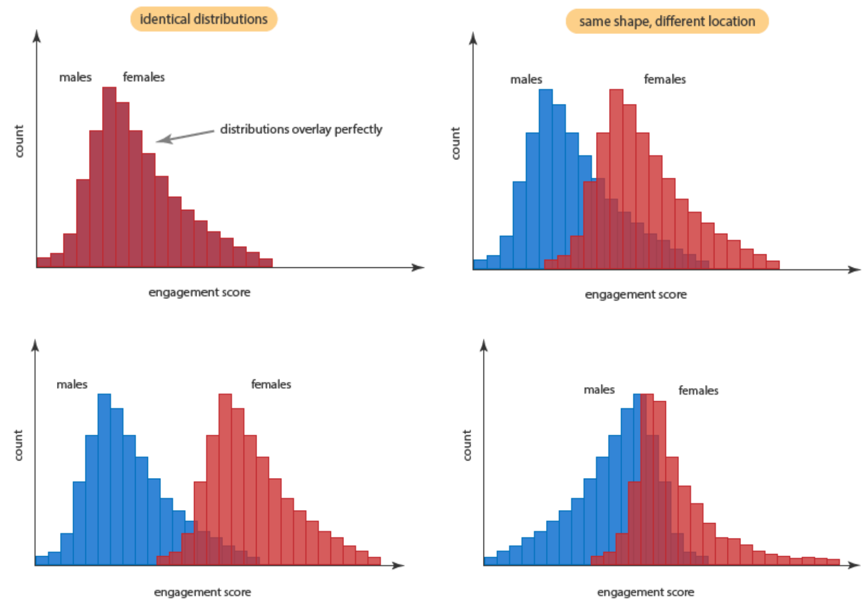
- Time measurement in C++
  - Different ways to measure the time
    - Classes
    - Functions…
  - A pragmatic way to measure the complexity of an algorithm
- Mean time is not enough
  - → Need of statistical analysis

Rationales

- Limitations to the theory
  - Duration of execution after the implementation
    - Can be different than what was expected
      - It depends on the size of the array (e.g. sorting)
      - It depends on the priorities of the thread, the class of the process…
- Simulations
  - Run n times an algorithm on some data
    - Data
      - Same exact data (same exact array)
      - Same type of data (different arrays but same size)

Rationales

- Question:
  - 2 algorithms f1 and f2, we want to compare the running time
  - What to do:
    - 1/ Theory: complexity
    - 2/ Practice: simulations
      - Run n trials with both algorithms
        - On n times $f1(x_i)$ and $f2(x_i)$ with $0<i<=n$
- Statistical evaluation
  - Rigorous



Constraints

- Statistical analysis
  - n values for f1, n values for f2
- Choice of the method
  - Normal distribution? Yes/No
  - Same number of observations for f1 and f2? Yes/No
  - f1 and f2 are evaluated when applied on the same input? Yes/No

Timer solutions

- RDTSC instruction
  - returns number of CPU cycles since the reset, 64 bit variable
  - very low-level
  - CPU cycles aren't steady time events: power saving, context switching…
- High performance timer on Windows → Acquiring high-resolution time stamps.
  - gives highest possible level of precision (<1us).
- GetTickCount: 10 to 16 milliseconds of resolution
- timeGetTime: (check MSDN website)
  - uses system clock (so the same resolution as GetTickCount)
  - but resultion can be increased up to even 1ms (via timeBeginPeriod)

Timer solutions

- std::chrono
  - timers from STL library
- system_clock
  - System time
  - Objects of class system_clock represent wall clock time from the system-wide realtime clock.
- steady_clock
  - monotonic clock
  - Objects of class steady_clock:
    - clocks for which values of time_point never decrease as physical time advances and for which values of time_point advance at a steady rate relative to real time. → the clock may not be adjusted.
- high_resolution_clock - highest possible resolution, multiplatform!
  - Warning:
    - might be alias for system or steady clock… depending on the system capabilities.

High performance timers

- Example:

```
#include "stdafx.h"
#include <windows.h>

int main()
{

    LARGE_INTEGER StartingTime, EndingTime, ElapsedMicroseconds;
    LARGE_INTEGER Frequency;

    QueryPerformanceFrequency(&Frequency);
    QueryPerformanceCounter(&StartingTime);

    // Activity to be timed

    QueryPerformanceCounter(&EndingTime);
    ElapsedMicroseconds.QuadPart = EndingTime.QuadPart - StartingTime.QuadPart;


    // We now have the elapsed number of ticks, along with the
    // number of ticks-per-second. We use these values
    // to convert to the number of elapsed microseconds.
    // To guard against loss-of-precision, we convert
    // to microseconds *before* dividing by ticks-per-second.
    //

    ElapsedMicroseconds.QuadPart *= 1000000;
    ElapsedMicroseconds.QuadPart /= Frequency.QuadPart;


    return 0;
}
```

Next slides

- For information only

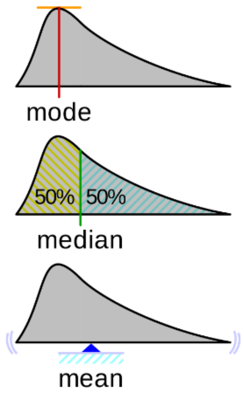
Statistical analysis

- **Many** methods
  - What to consider
    - Same number of observations in each group? Same variance? Analysis of pairs?
    - Parametric vs. Non Parametric
      - Example: Parametric: Anova
      - Example: Non parametric: Wilcoxon signed rank test
  - Example:
    - Student t-test
    - Wilcoxon signed rank test
      - non-parametric version of the two-sample t-test
        - No assumption on the distribution of the data
      - Sample size
      - Evaluation by pairs

Statistical analysis

- Some reminder
  - Mean

    arithmetic mean $$A = \frac{1}{n}\sum_{i=1}^{n} a_i = \frac{a_1 + a_2 + \cdots + a_n}{n}$$
  - Standard deviation $$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}, \text{ where } \mu = \frac{1}{N}\sum_{i=1}^{N} x_i.$$
  - Median
    - value separating the higher half of a data sample from the lower half
  - Mode
    - value that appears most often



Wilcoxon signed-rank test

- Statistical analysis
  - non-parametric statistical hypothesis test
    - comparing 2 related samples, matched samples, or repeated measurements on a single sample
    - Goal: to assess whether their population mean ranks differ
- N: sample size
  - 2 methods to test → 2N data points = 2N measurements
  - $x_{1,i}$: value for method 1 at measurement i
  - $x_{2,i}$: value for method 2 at measurement i
- Possibilities
  - H0
    - difference between the pairs follows a symmetric distribution around zero
  - H1
    - difference between the pairs does not follow a symmetric distribution around zero.

Wilcoxon signed-rank test

- For all the N measurements
  - Compute the distance between $x_{1,i}$ and $x_{2,i}$
    - Remove all the pairs where the difference is 0 → Nr = reduced sample size
  - Compute the sign between $x_{1,i}$ and $x_{2,I}$
    - - 1 if $x_{1,i} < x_{2,I}$
    - 0 if $x_{1,i} = x_{2,I}$
    - 1 if $x_{1,i} > x_{2,I}$
- Order all the pairs from smallest abs diff value to largest abs diff value
- Ranking: Ri rank in the list of pairs (smallest = 1)
- Test statistic W:

$$W = \sum_{i=1}^{N_r} [\mathrm{sgn}(x_{2,i} - x_{1,i}) \cdot R_i]$$

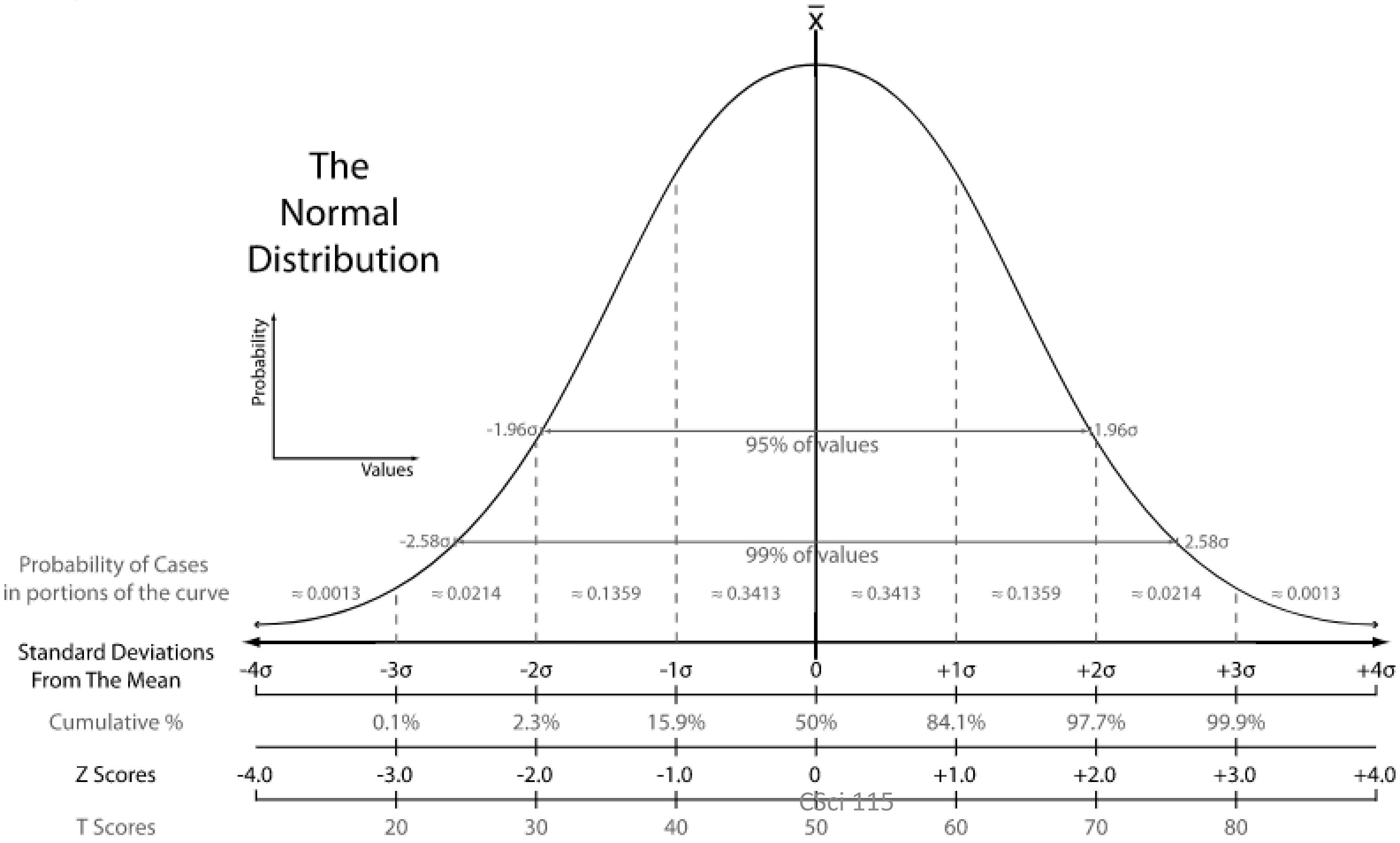
Wilcoxon signed-rank test

- Sum of all the ranks: n(n+1)/2
  - We know the max value when it s all positive or negative differences
    - Wmax=n(n+1)/2 ,Wmin=-n(n+1)/2
  - As many positive than negative → W = around 0
- W:
  - Expected value: 0
  - Variance: $\frac{N_r(N_r+1)(2N_r+1)}{6}$ standard_deviation=sqrt(variance)
- z-score= (samplevalue-mean)/standard_deviation

$$z = \frac{W}{\sigma_W}, \sigma_W = \sqrt{\frac{N_r(N_r+1)(2N_r+1)}{6}}$$

Statistical analysis

- Analysis of z



Mann–Whitney *U* test

- Another non-parametric test
  - a nonparametric test of the null hypothesis:
  - it is equally likely that a randomly selected value from **one sample** will be less than or greater than a randomly selected value from a **second sample**.
  - (no pairs)
- $H_{0:}$ null hypothesis, the distributions of both populations are equal
- $H_{1\,:}$ alternative hypothesis, the distributions are not equal.

Mann–Whitney *U* test

- Calculation:
  - n1: size of sample 1
  - n2: size of sample 2
  - Assign numeric ranks to all the observations
    - put the observations from both groups to 1 array, beginning with 1 for the smallest value.
  - Stat test value: U=min(U1,U2) where
    - U1=n1*n2+ n1(n1+1)/2 + R1
      - R1: sum of the ranks for group 1
    - U2=n1*n2+ n2(n2+1)/2 + R2
      - R2: sum of the ranks for group 2

Mann–Whitney *U* test

- From U to z-score $$z = \frac{U - m_U}{\sigma_U}$$
- Where $m_U = \frac{n_1 n_2}{2}$ and $\sigma_U = \sqrt{\frac{n_1 n_2 (n_1 + n_2 + 1)}{12}}$
- Special case with ties in ranks:
  - Example (3,4,4, 4,4,8)
    - → (1, 3.5, 3.5, 3.5, 3.5, 6) and the unadjusted rank is (1,2,3,4,5,6).

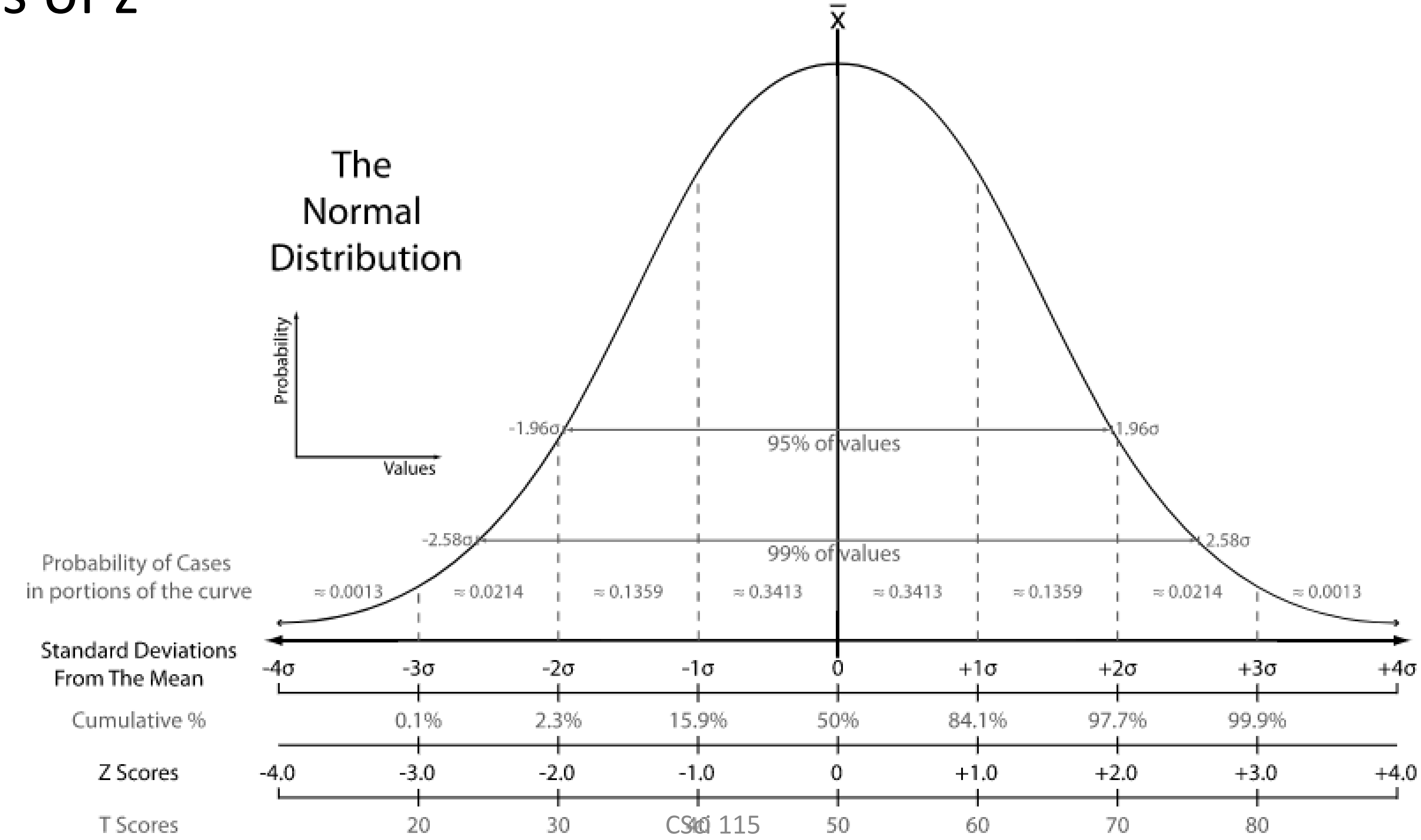
In this case:

$$\sigma_{\text{corr}} = \sqrt{\frac{n_1 n_2}{12} \left((n+1) - \sum_{i=1}^{k} \frac{{t_i}^3 - t_i}{n(n-1)} \right)}$$

With ti number of examples sharing rank i
and *k* is the number of (distinct) ranks.

Statistical analysis

- Analysis of z



Conclusion

- Time
  - Precious resource
- In new compilers/programming environment
  - It tells the time that is spent in each function automatically
  - Possibility to analyze potential bottlenecks
- Simulation and analysis of the results
  - Requirement of some statistical analysis
    - For a proper and rigorous analysis
      - **A single measurement alone tells nothing**
      - **The mean alone is not enough to represent the measurements**
- Questions?