

## CI2612: Algoritmos y Estructuras de Datos II

Blai Bonet

Universidad Simón Bolívar, Caracas, Venezuela

## Programa de estudio

© 2016 Blai Bonet

# Computer Science Curricula 2013

Curriculum Guidelines for  
Undergraduate Degree Programs  
in Computer Science

December 20, 2013

The Joint Task Force on Computing Curricula  
Association for Computing Machinery (ACM)  
IEEE Computer Society

## CS2013: Introduction

- “ACM and IEEE-Computer Society have a long history of sponsoring efforts to establish international curricular guidelines for **undergraduate programs** in computing on roughly a ten-year cycle, starting with the publication of Curriculum 68 over 40 years ago”
- “The last complete Computer Science curricular volume was released in 2001 (CC2001), and an interim review effort concluded in 2008 (CS2008)”
- “This volume, Computer Science Curricula 2013 (CS2013), represents a **comprehensive revision**”
- “The CS2013 guidelines include a redefined body of knowledge, a result of rethinking the **essentials necessary** for a Computer Science curriculum”

© 2016 Blai Bonet

## CS2013: Charter

The ACM and IEEE-Computer Society chartered the CS2013 effort with the following directive:

*To review the Joint ACM and IEEE-CS Computer Science volume of Computing Curricula 2001 and the accompanying interim review CS 2008, and develop a revised and enhanced version for the year 2013 that will match the latest developments in the discipline and have lasting impact*

*The CS2013 task force will seek input from a diverse audience with the goal of broadening participation in computer science. The report will seek to be international in scope and offer curricular and pedagogical guidance applicable to a wide range of institutions. The process of producing the final report will include multiple opportunities for public consultation and scrutiny*

## CS2013: Principles

1. Computer science curricula should be designed to provide students with the flexibility to work across many disciplines
2. Computer science curricula should be designed to prepare graduates for a variety of professions, attracting the full range of talent to the field
3. CS2013 should provide guidance for the expected level of mastery of topics by graduates
6. The size of the essential knowledge must be managed
8. CS2013 should identify the fundamental skills and knowledge that all computer science graduates should possess while providing the greatest flexibility in selecting topics

## CS2013: Body of knowledge

In Computer Science terms, one can view the Body of Knowledge as a specification of the content to be covered and a curriculum as an implementation

- Topics are identified as either “Core” or “Elective” with the core further subdivided into “Tier-1” and “Tier-2”
  - **A curriculum should include all topics in the Tier-1 core** and ensure that all students cover this material
  - **A curriculum should include all or almost all topics in the Tier-2 core** and ensure that all students encounter the vast majority of this material
  - **A curriculum should include significant elective material:** Covering only “Core” topics is insufficient for a complete curriculum
- The learning outcomes and hour counts in the Body of Knowledge provide guidance on the depth of coverage towards which curricula should aim.

### AL/Basic Analysis

[2 Core-Tier1 hours, 2 Core-Tier2 hours]

Topics:

[Core-Tier1]

- Differences among best, expected, and worst case behaviors of an algorithm
- Asymptotic analysis of upper and expected complexity bounds
- Big O notation: formal definition
- Complexity classes, such as constant, logarithmic, linear, quadratic, and exponential
- Empirical measurements of performance
- Time and space trade-offs in algorithms

[Core-Tier2]

- Big O notation: use
- Little o, big omega and big theta notation
- Recurrence relations
- Analysis of iterative and recursive algorithms
- Some version of a Master Theorem

## AL/Algorithmic Strategies

[5 Core-Tier1 hours, 1 Core-Tier2 hours]

An instructor might choose to cover these algorithmic strategies in the context of the algorithms presented in “Fundamental Data Structures and Algorithms” below. While the total number of hours for the two knowledge units (18) could be divided differently between them, our sense is that the 1:2 ratio is reasonable.

### Topics:

[Core-Tier1]

- Brute-force algorithms
- Greedy algorithms
- Divide-and-conquer (cross-reference SDF/Algorithms and Design/Problem-solving strategies)
- Recursive backtracking
- Dynamic Programming

[Core-Tier2]

- Branch-and-bound
- Heuristics
- Reduction: transform-and-conquer

## AL/Fundamental Data Structures and Algorithms

[9 Core-Tier1 hours, 3 Core-Tier2 hours]

This knowledge unit builds directly on the foundation provided by Software Development Fundamentals (SDF), particularly the material in SDF/Fundamental Data Structures and SDF/Algorithms and Design.

### Topics:

[Core-Tier1]

- Simple numerical algorithms, such as computing the average of a list of numbers, finding the min, max, and mode in a list, approximating the square root of a number, or finding the greatest common divisor
- Sequential and binary search algorithms
- Worst case quadratic sorting algorithms (selection, insertion)
- Worst or average case  $O(N \log N)$  sorting algorithms (quicksort, heapsort, mergesort)
- Hash tables, including strategies for avoiding and resolving collisions
- Binary search trees
  - Common operations on binary search trees such as select min, max, insert, delete, iterate over tree
- Graphs and graph algorithms
  - Representations of graphs (e.g., adjacency list, adjacency matrix)
  - Depth- and breadth-first traversals

[Core-Tier2]

- Heaps
- Graphs and graph algorithms
  - Shortest-path algorithms (Dijkstra's and Floyd's algorithms)
  - Minimum spanning tree (Prim's and Kruskal's algorithms)
- Pattern matching and string/text algorithms (e.g., substring matching, regular expression matching, longest common subsequence algorithms)

## AL/Advanced Data Structures Algorithms and Analysis

[Elective]

Many programs will want their students to have exposure to more advanced algorithms or methods of analysis. Below is a selection of possible advanced topics that are current and timely but by no means exhaustive.

### Topics:

- Balanced trees (e.g., AVL trees, red-black trees, splay trees, treaps)
- Graphs (e.g., topological sort, finding strongly connected components, matching)
- Advanced data structures (e.g., B-trees, Fibonacci heaps)
- String-based data structures and algorithms (e.g., suffix arrays, suffix trees, tries)
- Network flows (e.g., max flow [Ford-Fulkerson algorithm], max flow – min cut, maximum bipartite matching)
- Linear Programming (e.g., duality, simplex method, interior point algorithms)
- Number-theoretic algorithms (e.g., modular arithmetic, primality testing, integer factorization)
- Geometric algorithms (e.g., points, line segments, polygons. [properties, intersections], finding convex hull, spatial decomposition, collision detection, geometric search/proximity)
- Randomized algorithms
- Stochastic algorithms
- Approximation algorithms
- Amortized analysis
- Probabilistic analysis
- Online algorithms and competitive analysis

## CS2013: Más información

<http://www.cs2013.org>

## Instructor

Prof. Blai Bonet

Información de contacto:

- Oficina: MYS-215A
- Web: <http://www ldc usb ve/~bonet>
- Email: [bonet@ldc.usb.ve](mailto:bonet@ldc.usb.ve), [bonetblai@gmail.com](mailto:bonetblai@gmail.com)
- Horas de consulta: **a definir**
- Curso: <http://www ldc usb ve/~bonet/courses/ci2612>

© 2016 Blai Bonet

## Objetivos generales

- Garantías de desempeño de algoritmos y estructuras de datos en el peor caso y caso promedio
- Algoritmos recursivos e iterativos
- Pruebas de correctitud y análisis de desempeño
- Algoritmos randomizados y sus análisis
- Estructuras de datos elementales (cola, pila y lista enlazada)
- Estructuras de datos: colas de prioridad, diccionarios y árboles de búsqueda

© 2016 Blai Bonet

## Objetivos específicos

1. Nociones básicas
2. Complejidad en tiempo y espacio
3. Búsqueda lineal y binaria
4. Ordenamiento por inserción
5. Recurrencias y solución
6. Dividir y conquistar
7. Mergesort
8. Multiplicación de matrices
9. Algoritmo de Strassen
10. Probabilidad
11. Algoritmos randomizados
12. Análisis probabilístico
13. Heapsort y colas de prioridad
14. Quicksort y Quicksort randomizado
15. Cotas inferiores para ordenamiento
16. Cotas inferiores para mezcla
17. Ordenamiento en tiempo lineal
18. Cálculo de mediana y estadísticos
19. Pilas, colas y listas enlazadas
20. Apuntadores
21. Representación de árboles
22. Tablas de hash
23. Hashing universal
24. Hashing perfecto
25. Hashing cuckoo
26. Filtros de Bloom
27. Árboles binarios de búsqueda
28. Árboles rojo y negro

© 2016 Blai Bonet

## Cronograma (tentativo)

Sem.	Lunes	Miércoles
I	11/04: Introducción al curso.	13/04: Nociones básicas. Búsqueda lineal y binaria. Ordenamiento por inserción.
II	18/04: Recurrencias. Dividir y conquistar. Mergesort. Multiplicación de matrices.	20/04: Probabilidad. Análisis probabilístico.
III	25/04: Algoritmos randomizados. Freivalds.	27/04: Heapsort y colas de prioridad.
IV	02/05: Quicksort y quicksort randomizado.	04/05: Cotas para ordenamiento y mezcla.
V	09/05: Ordenamiento en tiempo lineal.	11/05: Cálculo de mediana y estadísticos.
VI	16/05: <b>Repaso.</b>	18/05: <b>No hay clases por viaje.</b>
VII	23/05: <b>Examen.</b>	25/05: Conjuntos dinámicos. ED elementales.
VIII	30/05: Tablas de hash. Encadenamiento.	01/06: Direccionamiento abierto.
IX	06/06: Hashing universal. Hashing perfecto.	08/06: Hashing cuco. Filtros de Bloom.
X	13/06: Árboles binarios de búsqueda.	15/06: Árboles rojo y negro.
XI	20/06: Árboles rojo y negro.	22/06: <b>Repaso.</b>
XII	27/06: <b>Examen.</b>	29/06: <b>Entrega de notas.</b>

© 2016 Blai Bonet

## Evaluación

Dos exámenes de 50% cada uno, en fechas tentativas de 23/05 (semana VII) y 27/06 (semana XII)

## Bibliografía

Cormen, Leiserson, Rivest y Stein. *Introduction to Algorithms*. MIT Press, 3ra edición

## Cursos previos de CI2612

Objetivo general:

*Al final del curso, el estudiante estará capacitado para diseñar e implementar un Tipo Abstracto de Datos (TAD), aplicar algoritmos y estructuras de datos clásicos en la solución de problemas computacionales y analizar la complejidad de algoritmos sencillos*

## Cursos previos de CI2612

Objetivos específicos:

1. Interpretar el comportamiento de un TAD en el estilo de especificación basado en modelos
2. Especificar formalmente un TAD utilizando el estilo basado modelos descritos mediante la lógica de primer orden y estructuras matemáticas
3. Utilizar técnicas de refinamiento de datos que permiten garantizar la consistencia de los modelos en la implementación de un TAD
4. Seleccionar estructuras de datos para implementar un TAD utilizando criterios de eficiencia en tiempo y recursos
5. Resolver problemas mediante el uso de algunos TADs y algoritmos clásicos
6. Representar información con tipos de datos estructurados recursivos
7. Diseñar programas estructurados que incluyan datos y funciones recursivas
8. Aplicar las nociones básicas del análisis de algoritmos en casos sencillos

## Comparación con cursos previos de CI2612

En este curso nos enfocamos en algoritmos y estructuras de datos eficientes (con garantías de desempeño) que se utilizan ampliamente en la actualidad

Nuestro énfasis es sobre el diseño, implementación y análisis de algoritmos correctos y eficientes

Los otros cursos hacen énfasis en la **especificación formal** de algoritmos y estructuras de datos, y en un **método deductivo de síntesis** a partir de especificaciones que deja de lado las garantías de desempeño, y que en la práctica tiene muy poco impacto