IFT2015 Structures de données
Liste détaillée de sujets

Detailed List of Subjects for the Final Examination

(English translation starts on Page 9)

Miklós Csűrös

Département d'informatique et de recherche opérationnelle
Université de Montréal

Automne 2012

F0 Introduction

Le but de ce document est de définir les compétences et connaissances requises dans le cours IFT2015 à l'examen final. L'examen constitue également la deuxième partie de l'examen pré-doctorale en structures de données. La connaissance des sujets marqués par ⋆ est exigée pour un «B/A-». Les sujets marqués par ⋆⋆ correspondent plutôt à un niveau «A+/A». Les notes marginales sont des références au livre de Sedgewick [*Algorithmes en Java*]. Les fichiers pour les notes de cours se trouvent à l'URL `http://www.iro.umontreal.ca/~csuros/IFT2015/A12/materiel/`.

F1 Principes d'analyse d'algorithmes

Références

▷ Sedgewick chapitre 2
▷ Notes sur les fondations : `notes01-recursion.pdf`.
▷ Notes sur l'analyse d'algorithmes : `notes04-analysis.pdf`.

Sujets

⋆ Principes de base : pire cas, meilleur cas, moyen cas. S§2.1,2.2,2.7

⋆ Croissance de fonctions communes : constantes, logarithmiques, polynomiales, exponentielles. Factorielle ($n!$), approximation de Stirling, nombres Fibonacci ($F_n = F_{n-1} + F_{n-2}$), nombres harmoniques ($H_n = \sum_{i=1}^{n} 1/i = \ln n + \gamma + o(1)$). S§2.3

⋆ Notion de temps amorti.

⋆⋆ Logarithme itéré, fonction d'Ackerman et son inverse.

⋆ Notation asymptotique : définitions de grand $O(f)$, petit $o(f)$, $\Theta(f)$ et $\Omega(f)$. Expressions avec $O()$ ou $o()$, règles d'arithmétique : $O(f) + O(g)$, $O(f) \cdot O(g)$. Relations avec la limite S§2.4

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c > 0 \qquad \Rightarrow f(n) = O(g(n));$$
$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \qquad \Leftrightarrow f(n) = o(g(n)).$$

Comparaison de croissances exponentielles, polynomiales et logarithmiques.

⋆ Techniques de preuve. Application directe de la définition pour démontrer $f = O(g)$ ou $f = o(g)$.

⋆ Récurrences typiques. S§2.5,2.6

$$
\begin{aligned}
f(n) &= f(n-1) + O(1) && f(n) = O(n);\\
f(n) &= f(n-1) + O(n) && f(n) = O(n^2);\\
f(n) &= f(n/2) + O(1) && f(n) = O(\log n);\\
f(n) &= f(n/2) + O(n) && f(n) = O(n);\\
f(n) &= 2f(n/2) + O(1) && f(n) = O(n);\\
f(n) &= 2f(n/2) + O(n) && f(n) = O(n \log n);
\end{aligned}
$$

⋆ Preuve par induction pour récurrences asymptotiques.
⋆⋆ Solution de récurrences par substitution de variables.

F2 Structures élémentaires et types abstraits

Références

▷ Sedgewick chapitres 3 et 4.
▷ Notes sur les listes : `notes02-linkedlist.pdf`.
▷ Notes sur les tableaux : `notes03-tableaux.pdf`.

Sujets

⋆ Blocs de construction pour programmes Java.	S§3.1
⋆ Tableaux.	S§3.2
⋆ Listes chaînées. Variations : listes circulaires, doublement chaînées. Sentinelles pour la tête et/ou la queue. Manipulation d'éléments sur la liste, insertion et suppression. Parcours d'une liste.	S§3.3,3.4
⋆ Notion d'un type abstrait, interface, implantation, client.	S§4.1
⋆ Types abstraits de files généralisées, piles et queues/files FIFO.	S§4.2,4.7
⋆ Implémentations de pile et de queue par tableaux ou listes chaînées. Efficacité d'implantations différentes (temps de calcul pour les opérations standardes). Débordement.	S§4.4,4.5,4.7

F3 Récursion et arbres

Références

▷ Sedgewick chapitre 5, section 4.3
▷ Notes sur les arbres : `notes05-trees.pdf`.

Sujets

⋆ Algorithmes récursifs. Diviser pour régner. S§5.1,5.2
⋆ Terminologie pour structures arborescentes : arbre k-aire, hauteur, niveau, profondeur. Implantation d'un arbre. S§5.4
⋆ Propriétés d'arbres binaires (relations entre le nombre de nœuds internes et externes ou la hauteur). S§5.5
⋆ Parcours d'un arbre : préfixe/préordre, infixe/dans l'ordre, postfixe/postordre, ordre de niveau. S§5.6
⋆ Algorithmes récursifs sur les arbres : calcul de taille, hauteur ou profondeur de sous-arbres. S§5.7

F4 Appartenance-union

Références

▷ Sedgewick chapitre 1.
▷ Notes sur Union-Find : `notes08-unionfind.pdf`.

Sujets

⋆ Problème de connexité, opérations d'appartenance-union. S§1.2
⋆ Structure Union-Find. Astuces : union rapide équilibrée, compression de chemin. S§1.3
⋆⋆ Coût amorti d'opérations : $O(\alpha(m, n))$ pour Union-Find avec union équilibrée et compression de chemin.

F5 File de priorité et tri par tas

Références

▷ Sedgewick sections 9.1–9.5.
▷ Notes sur tas binaire : `notes06-heap.pdf`.

Sujets

⋆ Type abstrait de file de priorité min-tas/max-tas : opérations insert, deleteMin ou deleteMax. Implantations par tableau ou liste chaînée. S§9.1,9.5
⋆ Arbre en ordre de tas. Manipulation du tas : nager/swim et couler/sink (heapisation montante et descendante). Tas binaire dans un tableau. Implantation de l'opération decreaseKey/increaseKey. S§9.2,9.3
⋆ heapify (établissement de l'ordre de tas dans un tableau). S§9.4
⋆⋆ Tas d-aire.
⋆ Tri par tas, son temps de calcul et usage de mémoire.

F6 Algorithmes sur graphes

Référence

▷ Sedgewick 3.7
▷ Notes sur l'arbre couvrant minimal et le plus court chemin : `notes09-acm.pdf`.

Sujets

⋆ Représentation d'un graphe : matrice d'adjacence et listes d'adjacence. S§3.7
⋆ Notion d'un arbre couvrant minimal. Principe de base des algorithmes : la règle bleue. Logique générale des algorithmes de Prim et de Kruskal, choix de structures de données. L'origin des temps de calcul $O(n \log n)$ et $O(m \log n)$.
⋆⋆ Analyse détaillé du temps de calcul des algorithmes. Avantages d'un tas d-aire ou Fibonacci dans l'algorithme de Prim.

F7 Méthodes de tri

Référence

▷ Sedgewick sections 6.1–6.4, 6.6, 3.4, 6.9, 10.2 ; chapitres 7, 8.
▷ Notes sur les tris : `notes10-tris.pdf`.
▷ Notes sur le tri rapide : `notes11-quicksort.pdf`.

Sujets

⋆ Terminologie : tri stable, tri interne et externe.	S§6.1
⋆ Tri par sélection et tri par insertion.	S§6.3,6.4
⋆ Performances des tris élémentaires (pire cas, meilleur cas, cas moyen).	S§6.6
⋆ Tri de liste chaînée.	S§3.4,6.9
⋆ Tri rapide : algorithme de base. Améliorations : partition par la médiane-de-trois, petits sous-fichiers.	S§7.1,7.4,7.5
⋆ Performances du tri rapide (pire cas, meilleur cas, cas moyen)	S§7.2,7.3
⋆⋆ Preuve de la performance moyenne $O(n \log n)$ du tri rapide.	
⋆ Fusion de tableaux.	S§8.1
⋆ Tri par fusion (descendant), sa performance.	S§8.3,8.4
⋆ Tri de clés binaires en temps linéaire.	§10.2

F8 Arbres binaires de recherche

Référence

▷ Sedgewick chapitres 12, sections 13.1, 13.3, 13.4
▷ Notes sur les arbres binaires de recherche : `notes12-abr.pdf`.
▷ Notes sur les arbres rouge-et-noir : `notes13-rn.pdf`.

Sujets

⋆ Type abstrait de la table de symboles. S§12.1,12.2
⋆ Recherche séquentielle et recherche binaire. S§12.3–12.5
⋆ Arbre binaire de recherche. Procédures fondamentales sur un ABR : recherche, insertion, suppression. Recherche de minimum ou maximum, successeur ou prédecesseur. S§12.6-12.9
⋆ Performance moyenne des opérations sur un ABR standard avec clés aléatoires. S§13.1
⋆ Notion d'un ABR équilibré. Maintenance d'équilibre : rotations simples et doubles.
⋆ ABR rouge et noir. Définition par rang (hauteur noire) ou coloriage ; équivalence des deux définitions. Coût des opérations de l'interface dans le pire cas. S§13.4
⋆⋆ Hauteur maximale d'un arbre rouge et noir.
⋆ Techniques de base sur les ABR rouges et noirs : promotion/rétrogradation, changement de couleur, rotation. Déroulement général d'une insertion ou suppression.
⋆⋆ Déroulement détaillé de l'insertion et de la suppression.
⋆ Les arbres 2-3-4, et leur équivalence avec les arbres rouges et noirs. Techniques de base sur les arbres 2-3-4 : décalage et découpage, leur relation aux rotations et promotions. S§13.3

F9 Tableaux de hachage

Référence

▷ Sedgewick chapitre 14.

▷ Notes sur le hachage : `notes14-hashing.pdf`.

Sujets

⋆ Notions de base pour tableaux de hachage : facteur de charge/remplissage, collisions. S§14.1

⋆ Fonctions de hachage : méthodes de la division et de la multiplication.

⋆⋆ Hachage universel.

⋆ Résolution de collisions par chaînage séparé. Coût moyen des opérations de l'interface (table de symboles) en fonction de la facteur de charge. S§14.2

⋆ Addressage ouvert : notion de sondage/test. Procédures de recherche et d'insertion avec addressage ouvert. Suppression paresseuse et hachage dynamique. Sondage linéaire, phénomène de grappe forte. Double hachage. S§14.3–14.6

⋆⋆ Coût moyen des opérations de l'interface avec sondage linéaire et double hachage en fonction de la facteur de charge.

◄ français English ►

E0 Introduction

Topics for a «B/A-» level are denoted by ⋆; ⋆⋆ denote somewhat more advanced topics for «A+/A» level. The margin notes refer to Sedgewick's *Algorithms in Java*. The class notes are available on the webpage `http://www.iro.umontreal.ca/~csuros/IFT2015/A12/materiel/`.

E1 Principles of algorithm analysis

References

▷ Sedgewick chapter 2
▷ Notes on foundations: `notes01-recursion.pdf`.
▷ Notes on algorithm analysis: `notes04-analysis.pdf`.

Topics

⋆ Basic principles : worst case, best case, average case. S§2.1,2.2,2.7
⋆ Growth of common functions : constants, logarithms, polynomials, exponentials. Factorial ($n!$), Fibonacci numbers ($F_n = F_{n-1} + F_{n-2}$), harmonic numbers ($H_n = \sum_{i=1}^{n} 1/i$). S§2.3
⋆ Notion of amortized cost.
⋆⋆ Iterated logarithm, Ackermann's function and its inverse
⋆ Asymptotic notation : definitions of big-Oh $O(f)$, small-oh $o(f)$, $\Theta(f)$, and $\Omega(f)$. Arithmetic expressions involving asymptotics, rules : $O(f)+O(g)$, $O(f)\cdot O(g)$. Connections to lim S§2.4

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c > 0 \qquad \Rightarrow f(n) = O(g(n));$$
$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \qquad \Leftrightarrow f(n) = o(g(n)).$$

Exponential, polynomial and logarithmic growth.
⋆ Proof techniques. Using the definitions to prove $f = O(g)$ or $f = o(g)$.
⋆ Basic recurrences. S§2.5,2.6

$$f(n) = f(n-1) + O(1) \qquad f(n) = O(n);$$
$$f(n) = f(n-1) + O(n) \qquad f(n) = O(n^2);$$
$$f(n) = f(n/2) + O(1) \qquad f(n) = O(\log n);$$
$$f(n) = f(n/2) + O(n) \qquad f(n) = O(n);$$
$$f(n) = 2f(n/2) + O(1) \qquad f(n) = O(n);$$
$$f(n) = 2f(n/2) + O(n) \qquad f(n) = O(n \log n);$$

⋆⋆ Variations on basic recurrences : "guessing" the asymptotics [iterated substitutions].
⋆ Proof by induction for asymptotic recurrences.
⋆⋆ Variable substitution for solving recursions.

E2 Elementary structures and abstract data types

References

▷ Sedgewick chapters 3 and 4
▷ Notes on lists: `notes02-linkedlist.pdf`.
▷ Notes on tables: `notes03-tableaux.pdf`.

Topics

⋆ Java building blocks.	S§3.1
⋆ Tables.	S§3.2
⋆ Linked lists. Variations : circular, doubly-linked lists. Sentinels for the head and/or tail. Manipulation of elements, insertion and deletion. List traversal.	S§3.3,3.4
⋆ Concept of an abstract type, interface, implementation, client.	S§4.1
⋆ Abstract types for stacks, queues and generalized queues,	S§4.2,4.7
⋆ Implementations of stack and queue by tables or linked lists. Running time for standard operations in different implementations. Overflow/underflow.	S§4.4,4.5,4.7

E3 Recursion and trees

Reference

▷ Sedgewick chapter 5, section 4.3
▷ Notes on trees: `notes05-trees.pdf`.

Topics

⋆ Recursive algorithms. Divide-and-conquer.	S§5.1,5.2
⋆ Terminology for tree structures : k-ary tree, height, level, depth. Tree implementations.	S§5.4
⋆ Mathematical properties of binary trees (relationships between number of internal and external nodes, height)	S§5.5
⋆ Tree traversal : preorder, inorder, postorder, level-order.	S§5.6
⋆ Recursions on trees : computing the size, height, or depth of subtrees.	S§5.7

E4 Union-find

References

▷ Sedgewick chapter 1.
▷ Notes on Union-Find: `notes08-unionfind.pdf`.

Topics

⋆ Problem of connectivity, union-find operations.	S§1.2
⋆ Union-Find data structure. Techniques : balanced fast union, path compression.	S§1.3
⋆⋆ Amortized cost per operations : $O(\alpha(m, n))$ for Union-Find with balanced trees and path compression.	

E5 Priority queues

References

▷ Sedgewick sections 9.1–9.5.

▷ Notes sur binary heap : `notes06-heap.pdf`.

Topics

⋆ ADT for priority queue : operations insert, deleteMin or deleteMax. Implementations by table or linked list. S§9.1,9.5

⋆ Heap order for a tree. Heap manipulation : swim and sink. Binary heap, its representation in a table. Implementation of decreaseKey. S§9.2,9.3

⋆⋆ d-ary heap.

⋆ heapify (linear-time construction of heap order in a table). S§9.4

⋆ Heapsort, its running time and memory usage. §9.4

E6 Graph algorithms

References

▷ Sedgewick 3.7

▷ Notes on minimum spanning tree and shortest path: `notes09-acm.pdf`.

Sujets

⋆ Graph representations by adjacency matrix and adjacency lists. S§3.7

⋆ Concept of a minimal spanning tree (MST). Basic principles of MST algorithms : the blue rule (adding minimum-weight edge in a cut). General logic of Kruskal's and Prim's algortithms, choice of data structures. Basic justification for $O(n \log n)$ and $O(m \log n)$ running times (n nodes, m edges).

⋆⋆ Detailed analysis of running time for Kruskal's and Prim's algorithms. Uses of d-ary or Fibonacci heap in Prim's algorithm.

E7 Sorting algorithms

References

▷ Sedgewick Sections 6.1–6.4, 6.6, 3.4, 6.9, 10.2 ; Chapters 7, 8.
▷ Notes sur sorting algorithms : `notes10-tris.pdf`.
▷ Notes sur quicksort : `notes11-quicksort.pdf`.

Topics

⋆ Terminology : stable sort, internal and external sort.	S§6.1
⋆ Insertion sort and selection sort.	S§6.3,6.4
⋆ Performance of elementary sorting algorithms (worst case, best case, average case).	S§6.6
⋆ Sorting a linked list.	S§3.4,6.9
⋆ Quicksort : basic algorithm. Improvements : pivoting by median-of-three, small subarrays.	S§7.1,7.4,7.5
⋆ Performance of quicksort (worst case, best case, average case).	S§7.2,7.3
⋆⋆ Proof of $O(n \log n)$ average running time for quicksort.	
⋆ Merging arrays.	S§8.1
⋆ Mergesort (top-down), its performance.	S§8.3,8.4
⋆ Linear-time sorting of binary keys.	§10.2

E8 Binary search trees

Reference

▷ Sedgewick chapters 12, sections 13.1, 13.3 and 13.4
▷ Notes on binary search trees: `notes12-abr.pdf`.
▷ Notes on red-black trees: `notes13-rn.pdf`.

Topics

⋆ Abstract data type of symbol table.	S§12.1,12.2
⋆ Sequential and binary search.	S§12.3–12.5
⋆ Binary search tree. Basic techniques : search, insertion, deletion. Searching for minimum or maximum, successor or predecessor.	S§12.6-12.9
⋆ Average performance of a standard BST with random keys.	S§13.1
⋆ Notion of a balanced BST. Maintaining the balance : simple and double rotations.	
⋆ Red-black tree. Definition by rank (black height) or coloring ; equivalence of the two definitions. Time complexity for operations in the worst-case.	S§13.4
⋆⋆ Maximum height of a red-black tree.	
⋆ Basic techniques for red-black trees : promotion/demotion, recoloring, rotations. General outline of insertion and deletion.	
⋆⋆ Detailed (case-by-case) steps in insertion and deletion.	
⋆ 2-3-4 trees, their equivalence with red-black trees. Basic techniques with 2-3-4 trees : shifting and splitting, relationship with promotions and rotations in red-black tree.	S§13.3

E9 Hash tables

Reference

▷ Sedgewick chapter 14.

▷ Notes on hashing: `notes14-hashing.pdf`.

Topics

⋆ Basic notions for hashtables : load factor, collisions.	S§14.1
⋆ Hash functions : division and multiplication methods.	
⋆⋆ Universal hashing.	
⋆ Collision resolution by separate chaining. Average-case performance with separate chaining as function of the load factor.	S§14.2
⋆ Open addressing : probe sequence. Search and insertion with open addressing. Lazy deletion, dynamic hashing. Linear probing, primary clustering. Double hashing.	S§14.3–14.6
⋆⋆ Average-case performance of linear probing and double hashing as function of the load factor.	