Luminy 98

D Goffinet
9 rue de la république
42000 St-Etienne
04 77 25 40 30
goffinet@anais.aurecvideo.fr

Arguments énergétiques en théorie des algorithmes:

Ce dont il s'agit

C'est un article de Eric Bach paru dans "The American Mathematical Monthly" de Novembre 97 (pp 831-837) qui m'a servi de support à une A.S.D.S. Ce texte peut être "apprécié" et critiqué à tous les niveaux, il me semble être une introduction idéale à une discussion sur toutes les questions liées aux preuves de programme.

Je vous en fournis ci-dessous ma traduction, si certains d'entre vous s'y connaissent, renseignez moi sur les risques que je prends concernant les copyrights, merci

Je précise que

- ci-dessous je ne fais *que* traduire (j'ai également essayé de traduire le mauvais anglais en mauvais français)
- j'ai ici l'original à la disposition des incrédules
- rendez vous à la fin pour une étude critique

Le texte

Mon objectif est de vous convaincre de ce que des arguments de type physique peuvent apporter des points de vue précieux dans l'étude du comportement des algorithmes

En particulier, divers résultats bien connus d'analyse combinatoire et de théorie des algorithmes sont faciles à prouver, à condition d'avoir le bon modèle physique. Ces démonstrations ont comme moteur l'idée simple et forte de la conservation d'énergie: dans un système physique clos, l'énergie totale ne change pas

Pour sûr, les idées physiques ont influencé l'algorithmique par d'autres chemins. Un exemple célèbre est le concept d'entropie qui a trouvé un large usage en théorie de l'information

Un exemple: vous savez sûrement qu'il faut $\ln_2(n!)$ comparaisons pour trier n valeurs. J'aimerais montrer ici que l'énergie est tout aussi utile et mérite une place permanente dans nos boites à outils. Par ailleurs, au niveau supérieur, les idées fondamentales de la conservation d'énergie amènent à la notion d'analyse amortie, notion à laquelle cet article peut être une introduction

longueur des chemins dans les arbres

Soit T un arbre binaire, chaque noeud a au plus deux enfants, un unique noeud nommé racine n'a pas de père. Dans l'étude des algorithmes de recherche, un rôle important est joué par la "longueur du chemin interne" définie par la somme sur les noeuds

$$\mathrm{C(T)} = \sum_{\mathrm{x}} \text{nombre d'arêtes entre x et la racine}$$

On considère aussi la "longueur du chemin externe" définie comme la somme étendue aux enfants manquant des noeuds de T

$$\mathrm{E(T)} = \sum_{\mathrm{y}} \text{nombre d'arêtes entre y et la racine}$$

. Traditionellement ces enfants manquant sont dits “noeuds externes”. Il est bien connu que l'on minimise ces fonctions en mettant toutes les feuilles de T sur deux profondeurs voisines ou si possible sur une seule. Nous allons montrer que cela découle immédiatement de considérations physiques

Imaginons que l'arbre pousse en partant de sa racine dans un champ gravitationel uniforme. Pensons à chaque noeud comme ayant la masse 1, donnons à la racine une énergie potentielle de 0, chacun des ses enfants ayant un potentiel de 1, ses petits enfants 2, etc. D'évidence C(T) est l'énergie potentielle totale de l'arbre qui sera minimisée en mettant autant de noeuds que possible à chaque niveau. D'où le résultat pour C(T)

Pour en déduire le résultat pour E(T), donnons aux noeuds externes une masse unité négative (Si cela vous gène, remplacez le champ gravitationnel par un champ électrique et utilisez des charges des deux signes). Imaginons maintenant un processus de démontage, chaque étape remplaçant l'une des feuilles les plus hautes de T et ses deux enfants externes par un noeud externe. Chaque étape de démontage augmente de 2 l'énergie de l'arbre. Si T a n noeuds il y a n étapes dans ce processus qui transforme un arbre d'énergie potentielle P(T)-E(T) en un arbre d'énergie potentielle 0. Donc, par conservation de l'énergie

$$\mathrm{E(T) = P(T) + 2n}$$

et tout arbre qui minimise P(T) minimise E(T). Pour des arbres d'arité d on peut prouver de façon analogue

$$\mathrm{E(T) = (d-1)P(T) + d \times n}$$

algorithmes de recherche optimaux

On peut appliquer le résultat sur les longueurs de chemins du § précédent aux algorithmes de recherche d'un clé parmi $x_1, ..., x_n$. Ces algorithmes sont composés de comparaisons pour savoir si x est plus grand, égal, ou inférieur à une autre clé x_j. Chacun de ces algorithmes détermine un arbre binaire de la manière suivante: la racine de l'arbre est étiquetée par la première clé comparée à x et les sous arbres gauches et droit sont déterminés par les questions utilisées selon que x sera inférieur ou supérieur à la racine

Considérons que la racine est au niveau 1, attribuons à chaque noeud de niveau i l'énergie potentielle i. Ainsi l'énergie potentielle d'un nœud est le travail (nombre de comparaisons) requis pour l'atteindre et, si tous les nœuds sont également susceptibles d'être cherchés, le travail moyen est l'énergie potentielle totale divisée par le nombre de nœuds. Ceci caractérise les algorithmes optimaux en moyenne: ce sont exactement ceux pour lesquels les feuilles sont sur deux niveaux adjacents, voire un. Pour $n=2^k-1$ il y a unicité d'un tel arbre et nous retrouvons la recherche dichotomique

Un problème voisin a été étudié par Huffmann et nous le reformulerons ainsi: nous avons des particules de masses positives $p_1, ..., p_n$, il faut les placer sur les nœuds externes d'un arbre binaire de manière à minimiser l'énergie potentielle totale. Comme l'arbre optimal ne va pas dépendre de notre unité de masse on peut supposer $\sum p_i = 1$. Ceci correspond à un jeu dans lequel nous devons identifier un élément parmi n susceptibles d'intervenir avec des possibilités proportionelles à $p_1..p_n$ en utilisant des questions de la forme “$x \in S$?”

Nous en tirons maintenant l'algorithme de Huffmann pour construire un arbre optimal qui détermine une stratégie de recherche minimisant l'espérance du coût de recherche. Dans tout arbre minimal T, les deux masses les plus faibles p et q doivent se trouver à la distance maximale de la racine, car sinon on pourrait échanger l'une d'entre elles avec la feuille la plus éloignée et diminuer le potentiel. Comme nous ne changeons pas le potentiel en déplaçant les masses horizontalement nous pouvons aussi bien en faire les fils d'un noeud interne x

Maintenant enlevons ces deux enfants et donnons leur masse totale (p+q) au noeud (maintenant externe) x. Cela fournit un nouvel arbre T'. Cette transformation relâche une quantité d'énergie égale à p+q donc la longueurs de chemins externes pondérés I_w vérifie:

$$I_w(T) = (p+q) + I_w(T')$$

récursivement on traite T' de même et cela résout le problème

Dans notre corollaire à l'algorithme de Huffmann nous avons supposé que chaque nœud était à distance unité de son père. Pourrait-on encore diminuer le potentiel si on autorisait ces longueurs à être fractionnaires? Pour éviter des évidences, nous ajouterons la contrainte suivante: si un noeud a deux fils aux distances l et r, alors $2^{-l} + 2^{-r} = 1$

L'algorithme qui en résulte est celui de Huffmann sauf qu'il met p et q aux distances $\ln_2((q+p)/p)$ et $\ln_2((q+p)/q)$ de leur père. Vous pourrez vérifier que l'arbre qui en résulte a pour énergie

$$H(p_1 ... p_n) = \sum -p_i \log_2 p_i$$

et que cela ne dépend jamais de l'énergie de l'arbre construit par l'algorithme de Huffmann

D'après ce résultat, tout arbre binaire avec des arcs de longueur 1 et des masses $p_1..p_n$ de somme 1 à ses nœuds externes a une énergie $\geq$ H Cette relation est à la base de nombreuses minorations en théorie de la complexité des algorithmes. Par exemple, pour prouver le résultat sur le tri cité dans l'introduction, nous prenons comme $\{p_i\}$ la distribution uniforme sur les n! permutations des données. Comme H est la fonction entropie bien connue, cela fournit un lien entre énergie et information. Pour d'autres tels liens [10]

calcul simultané du plus grand et du plus petit

Nous allons maintenant montrer que tout algorithme qui calcule le minimum et de maximum d'un ensemble de n clés par comparaisons doit effectuer $\frac{3n}{2} - 2$ tests dans le pire cas. Ce résultat est de Pohl [11]

Une méthode pour effectuer ce type de preuve est de fournir une règle selon laquelle les questions posées par l'algorithme pourront être utilisées pour bâtir un jeu de données "pire cas". L'algorithme sera alors puni par sa propre curiosité et la façon de faire cela est ce que l'on nomme une "stratégie d'adversaire"

Expliquons une telle stratégie en termes physiques. Chaque clé pourra avoir 4 états: V(ierge)=jamais comparée, G(rand)=au moins une fois Grand et jamais petit, P(etit)=au moins une fois petit et jamais grand, et E(liminé). On peut représenter tout cela dans le diagramme

$$\begin{tikzcd} & V \arrow[dl] \arrow[dr] & \\ P \arrow[dr] & & G \arrow[dl] \\ & E & \end{tikzcd}$$

Imaginez alors que les clés, comme des billes, dégoulinent dans ce diagramme. Attribuons à chaque bille une énergie égale à sa hauteur: 2 pour V, 1 pour P ou G et 0 pour E. L'énergie totale d'une configuration est la somme des énergies des billes. La stratégie d'adversaire est alors simple: quand une est posée on répond pour minimiser la perte d'énergie, mais en accord avec ce qui avait précédé. Supposons par exemple que, en calculant les max et min de x,y,z l'algorithme a déjà obtenu que $x < y$, soit une configuration d'énergie 4. On compare alors x à z. Si $x > z$ on passe à une énergie 3, si $x > z$ on passe à l'énergie 2. Comme les deux réponses sont cohérentes la stratégie adversaire choisira la première réponse, forçant ainsi à poser une troisième question.

Maintenant, l'algorithme doit passer de l'état dans lequel toutes les billes sont en V (énergie 2n) à un état dans lequel P et G ont unne bille, les autres étant en E (énergie 2). Chacune des comparaisons fait baisser l'énergie de 0,1 ou 2. Notons C_i le nombre de comparaisons faisant baisser l'énergie de i, on a donc

$$C_1 + 2C_2 = 2n - 2$$

mais M_2 est au plus n/2 (car on a n billes) car on peut toujours faire une chute d'énergie de 1 sauf pour les comparaisons V-V. Si on ajoute $-M_2 \geq -n/2$ à notre relation précédente, on a $M_1 + M_2 \geq 3n/2 - 2$ ce qui minore le nombre de comparaisons.

Notons aussi que ce modèle suggère un algorithme optimal. Prenons n pair (c'est plus simple) pour réduire au maximum l'énergie potentielle effectuons n/2 comparaisons V-V. On détermine ensuite le plus petit de P et le plus grand de G avec n/2-1 comparaisons à chaque fois

trouver le second plus grand

Nous allons maintenant prouver un théorème de Kislitsyn: tout algorithme opérant par comparaisons pour déterminer le second plus grand, doit utiliser au moins $n + \log_2(n) - 2$ comparaisons sur un certain jeu de données

Nous allons encore utiliser un modèle physique pour cet algorithme là: on pense aux clés comme à des particules, chacune ayant l'énergie 1. Quand deux clés sont comparées, la collision transfère toute l'énergie de l'une à l'autre, la restante tombe à 0. D'évidence une clé a une énergie positive ssi elle n'a jamais perdu de comparaison. Par conservation de l'énergie, la plus grande aboutit à l'énergie n, il a donc fallu au moins (n-1) comparaisons pour en arriver là car (n-1) autres clés ont du donner leur énergie. Nous allons donner à notre adversaire la tâche de rendre aussi dure que possible la recherche de la seconde clé. L'adversaire utilise la règle de Mathew "c'est ceux qui doivent l'avoir qui l'auront" de la façon suivante: quand on compare deux clés à énergie positive le gagnant est celui avec la plus grande énergie, on traite de même les matches nuls et les comparaisons de clés à énergie nulle

Supposons que la plus grande clé a été comparée à k autres clés. Elle avait donc une énergie au moins de n/2 avant la dernière comparaison, donc n/4 à la pénultième, etc. Comme son énergie initiale était 1 on a $1 \geq n/2^k$ donc $k \geq \log_2(n)$

Observons maintenant que (k-1) des clés comparées à la plus grande doivent avoir perdu au moins une autre comparaison pour que l'on puisse savoir qu'elles ne sont pas la seconde et les (n-k-1) autres clés doivent en avoir perdu au moins une. Le nombre total de comparaisons est donc minoré par

$$k + (k-1) + (n-k-1) \geq n + \log_2(n) - 2$$

comme le nombre de comparaisons doit être entier, on peut établir que c'est le meilleur possible

automates de recherche de motifs

Etudions maintenant l'algorithme Knuth-Morris-Pratt qui sert à dire si une chaîne donnée figure dans un texte. Supposons que nous cherchions AAAA alors l'algorithme construit un automate qui dans notre cas est

$$\text{départ} \rightleftarrows A \rightleftarrows A \rightleftarrows A \rightleftarrows A \rightleftarrows \text{gagné}$$

La recherche a lieu comme suit. Si nous sommes à l'état de départ nous lisons le caractère suivant du fichier et avançons à l'état de droite. Si nous sommes dans un état étiqueté par un autre caractère, nous regardons si le dernier caractère est le bon. Si oui nous lisons un autre caractère et avançons à droite, si non nous suivons la flèche gauche jusqu'à l'état précédent sans lire de nouveau caractère. L'arrivée à l'état succès indique que le motif a été trouvé. La propriété clé de cet algorithme est qu'il suffit de faire au plus 2n comparaisons de caractères pour fouiller un fichier de longueur n, cela une fois que l'automate a été construit. Nous allons maintenant prouver cela par un argument physique. On suppose que le motif a m caractères. Partant de l'état 0 nous numérotons les états de telle sorte que chaque comparaison réussie augmente le numéro d'état de 1. Les états sont donc 0,1..m+1. Disons que quand l'automate est dans l'état i il contient i unités d'énergie (si vous voulez, pensez à un ressort ou autre bidule qui se fait comprimer au fur et à mesure des reconnaissances de caractères) Chaque comparaison ratée relâche au moins une unité d'énergie car le numéro d'état diminue. D'un autre côté, au cours de la vie de l'algorithme on introduit au plus n unités d'énergie dans la machine parce que le seul moyen d'y mettre de l'énergie est de lire un nouveau caractère. Il y aura donc au plus n comparaisons ratées. Mais il y a au plus n comparaisons réussies car chaque réussite fait lire un nouveau caractère. Au total il y a donc au plus 2n comparaisons

On peut prouver que la machine peut être construite de la façon cannibale qui suit: cherchez le motif dans lui même. Plus précisément, si le motif est $p_1 ... p_m$ on le cherche dans $t = p_2 ... p_m$. L'indice du premier caractère du motif comparé au caractère du texte p_i détermine le recul. En sachant cela il faudra au plus 2m+2n comparaisons pour trouver un motif de longueur m dans un fichier de n caractères

un potentiel abstrait et l'algorithme d'Euclide

Tous nos exemples ont une particularité communne: nous affectons un nombre, l'énergie, à chacun des états d'un système et nous utilisons ce nombre pour tirer des conclusions concernant son évolution. Même sans regarder les détails du du système, mais en gardant seulement trace des changements énergétiques, on peut encore écrire la variation totale d'énergie comme la somme des changements d'énergie pendant l'histoire de l'algorithme

Relier ces changements énergétiques aux efforts de calcul concernés dans ces changements s'avère être un bon outil d'analyse des algorithmes. Dans la littérature on nomme cela "analyse d'amortissement" ou "méthode de la fonction potentielle". Voir [13]. Dans cette vision on n'est plus limité aux fonctions d'état issues de modèles physiques.

Je conclus cet article avec une preuve par fonction potentielle que le nombre d'opérations (binaires) utilisées dans l'algorithme d'Euclide est (à un facteur constant près) celui utilisé dans la multiplication ordinaire (résultat de Collins)

Ci dessous tous les log sont des $\log_2$, on note également O(f) toute fonction majorée par une constante positive$\times$f et $\lfloor x \rfloor$ la fonction partie entière

Discutons d'abord de notre modèle de coût. Si $u \in \mathbb{N}$ sa longueur binaire est $\text{long}(u)=\lfloor \log(u) \rfloor+1$. Pour $u \geq 2$ on a on a $\frac{1}{2}\text{long}(u) \leq \log(u) \leq \text{long}(u)$. Nous assignons à la multiplication de u par v le coût (long(u))(long(v)) et à la division euclidienne "$u=qv+r$ et $0 \leq r < v$ le coût (long(q))(long(v)). A des facteurs constants près, ce sont le nombres d'opérations utilisés par les algorithmes standards

Notre idée de l'algorithme d'Euclide est qu'il part avec $u \geq v > 0$ et qu'il remplace (u,v) par (v,u mod v) tant que $v > 0$, et qu'il stoppe et donne d quand il rencontre (d,0). Notre tâche est de prouver que le coût total d'Euclide est O(ln(u).ln(v)). L'idée de la preuve est d'attacher un potentiel ≥ 0 à chaque état de l'algorithme et de borner chaque coût de division par un multiple fixe du changement de potentiel. Comme on ne peut utiliser plus de potentiel que l'on en avait au départ, la différence de potentiel totale est bornée et c'est aussi le cas de notre coût de calcul total

Nous en sommes à choisir notre fonction potentiel. Il y a un équilibre naturel à conserver lors de ce choix: si on choisit une fonction simple, elle risque d'être difficile à relier au temps de calcul d'une étape. Une fonction pour laquelle cela sera facile pourra en revanche se révéler compliquée en cours de travail. Notre choix va être d'assigner à l'état (u,v) le potentiel ln(u).ln(v) Comme on finira avec v=0 ce cas devra être traité à part.

Considérons une étape de division de l'algorithme, autre que la dernière: $u=qv+r$ et $0 < r < v$ La différence de potentiel relative à ce pas est

$$\begin{aligned}\log(u).\log(v) - \log(v).\log(r) &= \log(v).\log(u/r) \\ &> \log(v).\log(u/v) \\ &\geq \log(v).\log(q)\end{aligned}$$

pour q=1 on n'en tire pas grand' chose, mais il y a alors une autre minoration: $u=v+r$ avec $0 < r < v$ donne $u/r \geq v/r + 1 \geq 2$ si bien que

$$\log(u).\log(v) - \log(v).\log(r) \geq \log(v).\log(u/r) \geq \log(u).\log(2)$$

A chaque étape (sauf la dernière) le potentiel diminue donc au moins de

$$\log(v)\log(\max(q,2)) \geq \frac{\text{long}(v).\text{long}(q)}{4}$$

Si maintenant notre algorithme consomme n pas, notons E_i le potentiel après le i-ème. Comme toutes les étapes (sauf la dernière) ont un reste > 0, le coût des (n-1) premiers pas est de

$$\sum_{i=1}^{n-1} \text{long}(q_i).\text{long}(v_i) \leq 4\sum(E_{i-1} - E_i) \leq 4E_0 \leq 4\text{long}(u).\text{long}(v)$$

la dernière division ne coûtera pas plus que long(u).long(v), le résultat est donc prouvé. En considérant la façon dont le potentiel varie toutes les deux étapes, on peut montrer que le nombre de pas de l'algorithme d'Euclide est O(log(u)). Résultat attribué à Lamé

mes remarques

longueur des chemins dans les arbres

Les "définitions" sont peu claires, pour y voir clair il faut faire un dessin, sur l'arbre syntaxique de a+b*(-c) on obtient P=9, E=21, la différence 12 est bien 2 fois 6. Les enfants manquants sont les deux fils qua a n'a pas, de même les deux absents pour b et c, enfin le - unaire a un déficit d'un fils.

La preuve par "conservation d'énergie" prétend utiliser ici les propriétés physiques des êtres dont il a utilisé les noms sans avoir fait de preuve. Une vraie preuve se fait facilement par récurrence sur la hauteur de l'arbre ou semi-structurellement: "T=T'+F" pour dire qu'un arbre est obtenu comme un sous-arbre avec une feuille en plus. Noter que (comme il l'utilise sans même l'affirmer), les enfants manquants des feuilles les plus hautes sont bien tous deux externes: c'est en effet évident

Remarquer que E(T)=P(T)+2n était malin, l'idée de la preuve par démontage suit bien l'idée de potentiel donnée par l'auteur. Dans le § qui précède, j'ai grogné pour bien peu

algorithmes de recherche optimaux

Le premier § n'est pas facile à lire, je l'ai compris (?) comme une formulation précise (?) de l'optimalité des tas dont j'ai alors réalisé que je vantais les avantages par rapport aux bêtes arbres binaires de recherche, sans avoir pensé qu'on pouvait envisager de se demander "de combien est-ce mieux?"

L'affirmation "Ceci correspond ..." me semble vraie dans un sens (on peut interpréter l'arbre ainsi), et fort douteuse dans l'autre. Le terme flou "correspond" était-il choisi pour cela?

L'affirmation "nous ne changeons pas ..." suppose le deux noeuds au même niveau. Si ce n'est pas le cas? La suite des phrases sous-entend (?) que le potentiel qu'il utilise serait la somme de $p_i \times$ hauteur. La conclusion "cela résoud le problème ..." (lequel) pourrait (?) signifier que ce qu'il y a de bien dans Huffmann c'est que les éléments les plus susceptibles d'être recherchés sont ceux qui sont le plus près de la racine.

Je n'ai rien compris aux nœuds à distances fractionnaires. Considère t'il l'arbre comme un graphe valué?

Que vient faire l'entropie ici? Il doit (?) avoir raison car ce la même définition se retrouve ("entropie binaire") dans l'excellent Cormen-Leiserson-Rivest; j'ai demandé aux physiciens autour de moi, ils n'ont pas vu le lien. Si vous savez, dites moi

Le dernier § du texte est encore un abus de mots, il me semble déduire des propriétés de la façon dont il nomme les choses

calcul simultané du plus grand et du plus petit

belle preuve, bien expliquée

trouver le second plus grand

A la fin il omet de dire comment on recherche effectivement ces deux plus grands, voici une solution:

on organise une coupe sans seeding (je ne sais pas comment on dit en français!): le vainqueur sera le plus grand, le second plus grand est simplement le plus grand de ceux qui ont perdu contre le vainqueur. Il faut bien sûr avoir mémorisé tous les résultats; sur un dessin, algorithme et preuve sont évidents

automates de recherche de motifs

C'est complètement fou! L'exemple est le pire que l'on ait pu choisir, les résultats finaux sont idiots, il parle peut-être de tout autre chose? Qui s'apellerait aussi KMP?

un potentiel abstrait et l'algorithme d'Euclide

Les deux premiers paragraphes décrivent bien l'idée de tout l'article

Les notations log et long sont faites exprès pour confuser le lecteur, ce sont mes traductions des log et ln de l'auteur

Sa vision d'Euclide, c'est très compliqué pour bien peu de chose! Son résultat final affirmé sans preuve est tout bête

quand on effectue deux divisions consécutives: a=bq+r puis b=rq'+r', je dis que $r' \leq b/2$ $\triangleright$si $r \leq b/2$ c'est évident puisque $r' < r$, et si r dépasse b/2 on a q'=1 et r'=b-r est bien inférieur à b/2 $\triangleleft$

encore mes remarques

Cet article plein de détails faux, de négligences molles ... contient pourtant des idées dont il est sûr que je me resservirai. Un taupin qui tiendrait de tels discours serait sûr de ne pas réussir à ses concours. Eliminons les gens à idées? Ou bien est-ce l'article d'un auteur ayant collationné des articles sur la complexité sans bien les comprendre?

C'est pourtant bon cette idée de consommation minimale d'énergie pour minimiser les coûts de calcul! même si ce n'est pas une idée originale de l'auteur il a eu le mérite de la faire connaître

toujours mes remarques

J'ai donné en A.S.D.S. introduction + longueur des chemins dans les arbres + recherche des extrêmes + les deux plus grands à l'un de mes très bons étudiants, le résultat fut un désastre, il a tout pris au premier degré. Nous (moi?) n'encourageons pas assez l'esprit critique