Algorithmes Additifs et Suites de Fibonacci.

TP-2, Module I41, Licence Informatique
université du sud Toulon-Var

Mars 2007

La suite de Fibonnacci F_n est la suite d'entiers positifs définie par les relations :

$$F_0 = 0, \quad F_1 = 1, \qquad \forall n \geq 2,\ F_n = F_{n-1} + F_{n-2}.$$

Nous avons vu en cours que F_n s'exprime à partir des deux racines l'équation $X^2 = X + 1$ qui sont $\phi \approx 1.618$ et $\hat{\phi} \approx -0.6$, par :

$$F_n = \frac{1}{\sqrt{5}}(\phi^n - \hat{\phi}^n)$$

En particulier, F_n croît exponentiellement F_n est équivalent à $\frac{1}{\sqrt{5}}\phi^n$ au voisinage de l'infini.

Par exemple $F_{10} = 55$, $F_{100} = 354224848179261915075$ et F_{1000} est un nombre de 289 chiffres décimaux : 43 466 557 686 937 456 435 688 527 675 040 625 802 564 660 517 371 780 402 481 729 089 536 555 417 949 051 890 403 879 840 079 255 169 295 922 593 080 322 634 775 209 689 623 239 873 322 471 161 642 996 440 906 533 187 938 298 969 649 928 516 003 704 476 137 795 166 849 228 875

1 Fibonacci récursif

1. Implanter une fonction récursive `long long fibrec(int n)` pourcalculer le n-eme terme de la suite de Fibonacci.

2. Soit $T(n)$ le temps de calcul. Faire des mesures de temps de calcul pour déterminer les meilleures constantes A et B tel que :

$$T(n) = AB^n.$$

2 Fibonacci itératif

1. Implanter une fonction itérative `long long fibiter(int n)` pour calculer le n-eme terme de la suite de Fibonacci.

2. Quelle est la forme du temps de calcul ?

3. Déterminer empiriquement la plus grande valeur de n pour laquelle votre fonction renvoie un résultat correct.

4. Déterminer empiriquement $\lim_{n\to\infty} \frac{F_{n+1}}{F_n}$.

5. Expliquer les résultats deux des questions précédentes.

3 Fibonacci sur les grands nombres

La figure (**??**) propose un extrait de programme en langage C pour calculer les termes de rang élevé de la suite de Fibonacci, les nombres étant représentés par des pointeurs sur des entiers.

1. Implanter la fonction `int digits(int n)` qui détermine la taille décimale de F_n.

2. Implanter la fonction `void add( int *s, int* a, int* b)` qui calcule dans s la somme des nombres a et b.

3. Implanter la fonction `void copier( int *d, int* s)` qui calcule dans d le nombre s.

4. Véfifier le bon fonctionnement de votre programme en calculant F_{1000}.

5. Déterminer une expression du temps de calcul en fonction de n.

```
int TAILLE;

int main( int argc, char *argv[])
{ int  n, *a, *b, *c;
       n = atoi( argv[1 ]);
  TAILLE = nbdigits( n );
  a = ( int* ) calloc( TAILLE, sizeof(int) );
  b = ( int* ) calloc( TAILLE, sizeof(int) );
  c = ( int* ) calloc( TAILLE, sizeof(int) );
  b[0] = 1;
  n = n - 2;
  while ( n > 0 ) {
    add( c, a, b );
    copier( a, b );
    copier( b, c )
    n = n - 1;
  }
  imprimer( b );
  free(a); free(b); free(c);
  return 0;
}
```

Figure 1: morceau de programme