PageRank

Résumé

Ce projet consiste à implanter l'algorithme de PageRank qui mesure la popularité des pages Internet. Plusieurs implantations de l'algorithme seront réalisées et leurs performances seront comparées. Les arguments de la ligne de commande permettront de choisir l'implantation à utiliser ainsi que les valeurs des paramètres de l'algorithme à utiliser.

Table des matières

1 Cahier des charges

1.1 PageRank

Brin & Page ont proposé en 1998 [1] de mesurer la popularité des pages Internet en les triant de la plus populaire à la moins populaire selon un ordre appelé *PageRank*. L'objectif de ce projet est de fournir les algorithmes nécessaires au calcul du *PageRank* pour un ensemble de pages Internet ou, plus généralement, un graphe orienté donné.

PageRank est exploité dans le moteur de recherche Google pour pondérer le résultat d'une requête dans le World Wide Web. PageRank est calculé à partir d'une métrique de popularité basée sur une idée très simple. Lors d'une recherche d'emploi, il peut-être demandé de fournir des lettres de recommandation. Ces lettres seront d'autant plus intéressantes que : (i) la personne qui vous recommande est respectable ; (ii) et que cette personne n'est pas connue pour écrire trop de lettres de recommandation.

Une personne respectable est une personne reconnue comme telle par d'autres personnes respectables, qui elles-mêmes sont reconnues par d'autres personnes, etc. L'analogie avec la qualité d'une page Internet se fait à l'aide des hyperliens : une page Internet est respectable si beaucoup d'autres pages la référencent, et d'autant plus si ces pages sont très référencées également par d'autres, etc. On peut donc définir le poids $r(P_i)$ d'une page P_i comme la somme du poids des pages qui la référencent, récursivement :

$$r(P_i) = \sum_{P_j \in \mathcal{P}_i^{IN}} r(P_j)$$

où $\mathcal{P}_i^{IN}$ est l'ensemble des pages qui référencent P_i. Néanmoins, il faut ajuster la popularité des P_j à leur comportement. Si les pages de $\mathcal{P}_i^{IN}$ référencent beaucoup de pages, elles ont moins de valeur. Pour prendre cela en compte, on pondère le poids de chaque P_j par le nombre de pages $|P_j|$ qu'elle référence au total (ce qui correspond au degré sortant de P_j) :

$$r(P_i) = \sum_{P_j \in \mathcal{P}_i^{IN}} \frac{r(P_j)}{|P_j|}$$

où $|P_j|$ est le nombre d'hyperliens de la page P_j.

Pour calculer les poids des pages $r(P_i)$, il est possible d'appliquer un algorithme très simple. Soit $r_k(P_i)$ la valeur du poids de la page P_i à l'itération k :

$$r_{k+1}(P_i) = \sum_{P_j \in \mathcal{P}_i^{IN}} \frac{r_k(P_j)}{|P_j|}$$

On initialise chaque valeur avec $r_0(P_i) = 1/N$, où N est le nombre de pages web à classer. Les indices des pages sont prises dans l'ensemble $[0, 1, .., N-1]$. Quand les valeurs $r_{k+1}(P_i)$ sont très proches de $r_k(P_i)$, il y a convergence vers la valeur de $r(P_i)$.

1. Brin et Page, *The anatomy of large-scale hypertextual Web search engine.* In Computed Networks and ISDN Systems, 33 :107-17, 1998.

Il reste alors à classer l'ensemble des pages web par ordre décroissant de leur poids $r(P_i)$. Le rang d'une page web dans cet ordre est appelé le *PageRank*. Ainsi, une page web de rang faible (*PageRank* petit) est une page importante pour le graphe des pages web. Plus le rang d'une page est faible, plus la probabilité est importante qu'une marche aléatoire lancée sur n'importe quelle autre page passe par cette page web.

1.2 Graphe orienté et pages web

L'ensemble des pages Internet et des hyperliens peut être modélisé par un graphe (ou réseau) orienté. Un graphe se compose d'un ensemble de nœuds (sommets) et d'arcs (liens ou arrêtes) qui relient deux nœuds selon une direction, le premier est l'origine de l'arc, le second la cible.

Dans notre cas, les pages de l'Internet sont modélisées par un graphe. Les pages web sont les nœuds du graphe et les hyperliens sont les liens orientés. Un exemple de graphe formé par 6 pages web est représenté à la Figure 1. Les pages web P_0 et P_1 sont liées par un lien orienté qui va de P_0 à P_1. Ce lien représente l'hyperlien présent sur la page P_0 qui permet d'accéder à la page P_1.

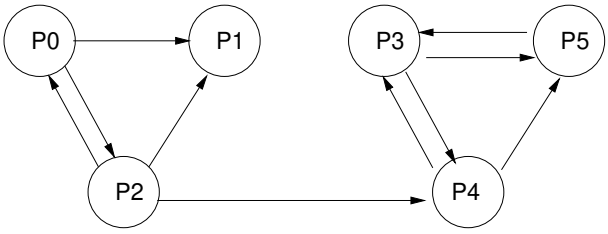

FIGURE 1 – Graphe de 6 pages web. Les hyperliens sont orientés : la page à l'origine de l'arc comporte un hyperlien vers la page cible de l'arc.

On notera que le calcul de *PageRank* peut se faire sur n'importe quel graphe orienté, que les nœuds représentent des pages web, des commutateurs, des véhicules...

1.3 Calcul matriciel du PageRank

On peut représenter le calcul de *tous* les poids des pages web au rang $k+1$ par une multiplication matricielle :

$$\pi_{k+1}^T = \pi_k^T \cdot H \qquad \text{si } k > 0 \tag{1}$$

$$\pi_0^T = (1/N, 1/N, ..., 1/N) \tag{2}$$

où :

— $\pi_k^T = (r_k(P_0), ..., r_k(P_i), ..., r_k(P_{N-1}))$ est un vecteur ligne [2] dont la composante i et le poids de la page P_i au rang k, et

— H est une matrice d'adjacence pondérée qui comporte, pour chaque ligne i, la valeur $1/|P_i|$ s'il existe au moins un lien sortant de la page P_i vers la page P_j, et 0 sinon.

2. Notation : x^T est la transposée du vecteur colonne x.

On notera que la somme des éléments d'une ligne de H est égale à 1.

Si on considère la représentation des liens orientés entre les 6 pages web de la figure 1, on obtient la matrice H de la figure 2.

$$H = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad S = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ \mathbf{1/6} & \mathbf{1/6} & \mathbf{1/6} & \mathbf{1/6} & \mathbf{1/6} & \mathbf{1/6} \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

FIGURE 2 – Matrices H (à gauche) et S (à droite) du graphe de la figure 1.

Pour bien répartir les poids et garantir la convergence de l'algorithme, Brin et Page ont introduit plusieurs modifications à la matrice H. La première transforme la matrice H en une matrice S, et la seconde transforme la matrice S en la matrice G (appelée matrice de *Google*).

Matrice S Dans l'exemple de la Figure 1, la page P_1 ne contient pas d'hyperlien (lien sortant) : c'est un puits ou cul-de-sac [3]. Dans la matrice H, il n'existe donc pas de valeur non nulle associée à P_1. Pour obtenir des poids raisonnables, on donne la valeur $1/N$ aux éléments $H(i, j)$ des nœuds en cul-de-sac P_i. La matrice résultante S est représentée dans la Figure 2.

Matrice de Google G En réalité, le calcul des poids des pages menant au PageRank se fait avec la matrice de Google définie par :

$$G = \alpha \cdot S + \frac{(1-\alpha)}{N}\mathbf{e}\mathbf{e}^T \quad \alpha \in [0, 1] \tag{3}$$

G est la somme de la matrice $\alpha \cdot S$ et d'une matrice dont tous les termes valent $(1 - \alpha)/N$. Dans (3), la notation $\mathbf{e}$ représente un vecteur dont tous les éléments valent 1. La pondération α est appelée le *dumping factor*. Plus sa valeur est proche de 1, plus la convergence du calcul est lente. Plus la valeur est faible, plus le calcul est rapide mais moins les poids prennent en compte la topologie du graphe. En pratique, $\alpha = 0.85$ offre un bon compromis entre rapidité de calcul et qualité des résultats.

$$G = \begin{bmatrix} 0.025 & 0.45 & 0.45 & 0.025 & 0.025 & 0.025 \\ 0.16666667 & 0.16666667 & 0.16666667 & 0.16666667 & 0.16666667 & 0.16666667 \\ 0.30833333 & 0.30833333 & 0.025 & 0.025 & 0.30833333 & 0.025 \\ 0.025 & 0.025 & 0.025 & 0.025 & 0.45 & 0.45 \\ 0.025 & 0.025 & 0.025 & 0.45 & 0.025 & 0.45 \\ 0.025 & 0.025 & 0.025 & 0.875 & 0.025 & 0.025 \end{bmatrix}$$

FIGURE 3 – Matrice de Google G du graphe à 6 pages web pour α **= 0.85**

3. dangling node, en anglais

Pour le calcul du terme suivant du vecteur poids, on remplace l'équation (1) par l'équation (4) suivante : on utilise G au lieu de H.

$$\pi_{k+1}^T = \pi_k^T \cdot G \tag{4}$$

1.4 Représentation textuelle d'un graphe

Pour représenter textuellement un graphe, nous choisissons la syntaxe suivante. Sur une première ligne est indiqué le nombre de nœuds, 6 dans l'exemple figure 1. Chaque ligne suivante représente un arc avec d'abord le nœud origine puis le nœud cible séparés par un ou plusieurs blancs. Chaque nœud est un entier i avec $0 \leq i < N$ où N est la taille du graphe.

La figure 4-(a) présente le fichier `sujet.net` qui correspond au graphe de la figure 1. Il s'agit bien d'un réseau de 6 nœuds. La ligne 2 indique qu'il existe un arc (un hyperlien) du nœud 0 vers le nœud 1.

Notons qu'une même page peut référencer plusieurs fois la même page et qu'il n'y a pas d'ordre imposé sur les arcs dans le fichier `.net`. Ainsi, le fichier de la figure 4-(d) est un autre fichier qui représente aussi le graphe de la figure 1.

```
6
0 1
0 2
2 0
2 1
2 4
3 4
3 5
4 3
4 5
5 3
```

sujet.net

Fichier Graphe
(a)

```
3
5
4
1
2
0
```

sujet.pr

Fichier PageRank
(b)

```
6 0.85000000000000 150
0.34870368521482
0.26859608185466
0.19990381197332
0.07367926270376
0.05741241249643
0.05170474575702
```

sujet.prw

Fichier poids
(c)

```
6
5 3
0 2
2 1
2 0
2 4
0 1
0 2
3 4
3 5
4 5
4 3
3 5
```

sujet-alternative.net

Fichier Graphe (autre)
(d)

FIGURE 4 – Un fichier `.net` (a) et (d) qui décrit le graphe de la figure 1, le fichier `.pr` (b), classement des nœuds suivant l'algorithme PageRank et le fichier `.prw` (c), poids des nœuds. L'écriture des réels a été obtenue en utilisant la procédure `Put (x, Fore => 1, Exp => 0)`, où `Fore` indique que d'utiliser 1 position au moins pour la partie entière et `Exp` le nombre de chiffre pour l'exposant (donc sans exposant ici). La précision de calcul est ici de 15 chiffres (`Long_Float`).

1.5 Fichiers résultats

L'algorithme de *PageRank* doit produire deux fichiers : le fichier *PageRank*, extension `.pr`, et le fichier des *poids*, extension `.prw`. Ces deux fichiers sont obtenus après un tri décroissant sur

le poids des nœuds. Le fichier *PageRank* liste l'identifiant des nœuds par ordre décroissant de poids, et le fichier *poids*, liste la valeur de poids des nœuds. La première ligne du fichier *poids* renseigne, dans l'ordre, le nombre total de nœuds, la valeur d'α et l'indice du vecteur poids (valeur de k).

1.6 Précision des calculs

Les calculs pourront être réalisés avec différentes précisions. En Ada, on peut choisir la précision des nombre réels avec `digits PRECISION`[4] comme dans :

```
type T_Double is digits PRECISION;
```

où `PRECISION` doit être une constante entière connue au moment de la compilation qui détermine le nombre (minimal) de chiffres significatifs du nombre. Par exemple, si on utilise `digits 2` (2 chiffres significatifs), alors `0.1264` sera représenté comme `0.13`, `3.1415` comme `3.1` et `543.21` comme 540. Notez que Ada utilisera le type des réels disponible sur le machine suffisant pour la précision demandée (et donc certainement avec une précision plus grande que celle qui est demandée).

Pour lire et écrire des objets du type `T_Double`, on instanciera `Ada.Text_IO.Float_IO` comme suit. On aura alors accès aux opérations `Put` et `Get`.

```
package Double_IO is new Ada.Text_IO.Float_IO (T_Double);
use Double_IO;
```

Il est possible de définir un paramètre générique qui représente la famille de types Réels en Ada. Pour cela, on utilise la déclaration suivante :

```
generic
    type T_Reel is digits <>;    --! type réel de précision quelconque
```

Le module pourra être instantié avec n'importe quel type réel (`Float`, `Long_Float`, `Long_Long_Float` ou son propre type réel comme `T_Double`). Le paramètre de généricité étant un type réel, on peut donc utiliser les opérateurs arithmétiques usuels (`+`, `*`, `/`...), ou encore travailler directement avec des constantes littérales (`1.0`, `0.0`...).

Remarque : Cet aspect du sujet ne sera traité que quand tous les autres aspects seront réalisés.

1.7 Le programme PageRank

L'objectif de ce projet est de fournir un programme capable de calculer, pour un graphe donné, le *PageRank* et le poids de chaque nœud du graphe. L'algorithme (équations (4) et (2)) sera implanté en utilisant la matrice de Google G (équation (3)).

Cet algorithme sera implanté de plusieurs manières :

1. en calculant la matrice G. C'est l'approche *matrice pleine*.

4. `https://en.wikibooks.org/wiki/Ada_Programming/Types/digits`

2. en exploitant le fait que la matrice H est très creuse. C'est l'approche *matrice creuse*.
   Google estime qu'actuellement, il y a environ 56 milliards de pages Internet[5]. Ainsi, le nombre de lignes de G est très grand. Il en est de même pour la taille de π. En revanche, le nombre de valeurs $G(i, j)$ non nulles pour une page P_i est d'environ 10. En effet, en moyenne, une page web ne possède qu'une dizaine d'hyperliens. La matrice H est donc très « creuses ». L'idée est donc de ne pas calculer explicitement la matrice G et de s'appuyer sur le fait que la matrice H a de nombreuses valeurs nulles pour limiter les données stockées et organiser les calculs pour limiter les opérations à réaliser et ainsi améliorer l'efficacité de l'algorithme en espace mémoire et temps de calcul.

1.8 Paramètres de la ligne de commande

Le programme ne devra faire aucune interaction avec l'utilisateur. Son comportement sera exclusivement défini au moyen des paramètres de la ligne de commande. La forme générale pour appelé le programme sera la suivante :

```
./pagerank [options] reseau
```

où `reseau` correspond au fichier `.net` à analyser et où les options, optionnelles, sont les suivantes :

`-A <valeur>` Définir la valeur de α. La valeur doit être comprise entre 0 et 1 au sens large. Si l'option `-A` n'est pas précisée, on utilisera la valeur 0.85.

`-K <valeur>` Définir l'indice k du vecteur poids à calculer, π_k, grâce à l'algorithme *PageRank*. La valeur doit être un entier positif. Sa valeur par défaut est 150.

`-E <valeur>` Définir une précision (un epsilon) qui permettra d'interrompre le calcul du *PageRank* si le vecteur poids est à une distance du vecteur poids précédent strictement inférieure à epsilon. La valeur est un nombre réel positif. Sa valeur par défaut est 0.0 (donc désactivé).

La valeur de l'option `-K` limitera le nombre de termes de π_k calculés. Ceci garantira la terminaison du programme.

La distance entre deux vecteurs sera la plus grande des distances entre les composantes des deux vecteurs (valeur absolue de la différence des composantes).

`-P` Choisir l'algorithme avec des matrices pleines (la matrice G est calculée).

`-C` Choisir l'algorithme avec des matrices creuses (la matrice G n'est pas calculée). C'est l'algorithme qui est mis en œuvre par défaut.

`-R <prefixe>` Choisir le préfixe des fichiers résultats, par défaut `output`. Les fichiers produits seront donc `<prefixe>.pr` pour le classement des nœuds et `<prefixe>.prw` pour les poids.

Voici quelques utilisations possibles du programme.

1. `./pagerank sujet.net`
   Analyser le fichier `sujet.net` en version matrice creuse, avec calcul de l'indice 150 et une valeur de α à $0,85$, epsilon valant 0.0.

5. `http://www.worldwidewebsize.com/`

2. `./pagerank -P -A 0.90 -K 20 exemple2.net`
   Analyser le fichier `exemple2.net` avec une valeur de α à $0,90$, avec matrice pleine, calcul de l'indice 20 avec epsilon valant 0.0.
3. `./pagerank -E 0.001 -K 20 -P -E 0.0001 -A 0.70 -C exemple3.txt`
   Analyser le fichier `exemple3.txt` avec une valeur de α à $0,70$ avec des matrices creuses, et une précision de 0.0001, en s'arrêtant au pire sur le vecteur poids π_{20}. On constate que c'est la dernière occurrence d'une option qui l'emporte sur les précédentes.
4. `./pagerank -E 0.01 -K 30 -C -A 1 exemple4.txt`
   Analyser le fichier `exemple3.txt` avec une valeur de α à 1 en utilisant la version creuse, un epsilon de 0.01 et un calcul de π_{30} au pire (si la précision souhaitée n'est pas atteinte avant).
5. `./pagerank -R exemple5 -K 10 -R exemple5-K10  exemple5.txt`
   Dans les exemples précédents, le préfixe des fichiers résultats n'étaient pas précisés et le préfixe `output` était utilisé. Ici, les fichiers produits seront donc `exemple5-K10.pr` `exemple5-K10.prw` (seule la dernière valeur de `-R` est prise en compte).
6. `./pagerank -K 500 -K -10 -A 2.5 -C exemple6.txt`
   Le programme signalera une erreur sur la valeur qui suit l'option `-K` car elle doit être positive. On pourra s'arrêter à la première erreur ou signaler les erreurs suivantes. Ici la valeur qui suit `-A` doit être comprise entre 0 et 1.

1.9 Exemples de graphe

Plusieurs exemples de graphe seront donnés.

Aucun exemple de graphe n'est donné pour valider la robustesse de votre programme. Il sera utile d'en définir.

1.10 Mesurer l'efficacité du programme

Le but étant de pouvoir traiter des graphes de grande taille, l'efficacité du programme en version creuse sera importante. Pour mesurer le temps d'exécution d'un programme, on peut utiliser sa montre mais cette mesure n'est pas satisfaisante car plusieurs programmes peuvent s'exécuter simultanément sur la machine. De plus, on ne sait pas quelles parties du programme prennent le plus de temps à s'exécuter. Une meilleure approche est donc d'utiliser un outil de profilage.

Comme les graphes visés sont de très grande taille, nous vous demandons de mesurer, pour les différentes données de test, le temps d'exécution de votre programme sur une machine du centre informatique, en moyenne.

On peut utiliser la commande `gprof` qui mesure le temps passé dans les différents sous-programmes de votre programme. Les pages suivantes expliquent son utilisation :

— `https://gcc.gnu.org/onlinedocs/gnat_ugn/Profiling-an-Ada-Program-with-gprof.html`

— `http://www.thegeekstuff.com/2012/08/gprof-tutorial`.

On pourra différentier le temps passé dans le calcul du PageRank, celui passé à trier le vecteur final et celui passé à charger le graphe.

2 Contraintes

1. Le projet sera réalisé en équipes de 2 étudiants du même groupe de TD, éventuellement du même CM. Il y aura au plus une équipe de 3 dans chaque CM.
2. Le programme doit utiliser le langage Ada vu en PIM.
3. Toutes les notions vues en cours, TD ou TP peuvent (doivent !) être utilisées.
4. Le programme ne doit faire aucune interaction avec l'utilisateur. Seule la ligne de commande est utilisée.
5. Le programme devra être robuste concernant la ligne de commande et la structure du fichier d'entrée qui décrit le graphe. Si un problème est identifié par le programme, un message clair sera affiché pour en informer l'utilisateur.

3 Livrables

Les **documents à rendre** (via Git/Gilab) sont :

1. Les sources de votre projet, y compris les programmes de test et les exemples de graphes que vous aurez créés. Il ne faut pas rendre ceux qui ont été founis. Pour chaque programme de test et chaque graphe, il faut préciser ce qu'il permet de vérifier.
2. Le rapport qui doit comporter au moins :
   - une page de garde avec titre, année, numéro d'équipe, membres de l'équipe, etc.
   - un résumé qui décrit le contenu du rapport
   - une courte introduction qui présente le problème traité et le plan du document
   - l'architecture de l'applications (modules et dépendances entre modules)
   - les principaux choix réalisés, structures de données et algorithmes
   - l'explication de la manière dont le programme et les modules ont été mis au point
   - les durées d'exécution du programme en fonction des implantations (pleine et creuse), sur les exemples fournis et une analyse des résultats constatés
   - les difficultés rencontrées et les solutions adoptées en justifiant vos choix (en particulier quand vous avez envisagé plusieurs solutions)
   - les grilles d'évaluation des raffinages et du code complétées (cf Moodle).
   - une matrice avec en ligne les modules/sous-programmes, en colonne les activités (spécifier, programmer, tester, relire) avec dans les cases les initiales de l'équipier qui a réalisé l'activité pour la fonction considérée.

— une conclusion expliquant l'état d'avancement du projet (ce qui est fait, ce qui a été commencé, ce qui n'a pas été fait) et les perspectives d'amélioration / évolution
— en annexe, les apports personnels tirés de ce projet

Remarque : Le rapport ne devrait pas dépasser 12 pages, hors annexes.Il s'agit de la limite supérieure. Il n'est donc pas nécessaire de l'atteindre !

Les raffinages seront dans le document partagé et n'ont pas à être repris dans le rapport final. Le document des raffinages sera considéré comme une annexe implicite de votre rapport.

3. Le script de la démonstration (fichier démonstration.pdf) qui reprend ce qui sera fait pendant la démonstration.
4. Le support de présentation (presentation.pdf) qui sera utilisé lors de la présentation orale.

4 Échéances

Voici les **principales dates** (échéance à minuit) du projet :

— Mardi 21 novembre : **Publication du sujet du projet**.
— Jeudi 23 novembre : **Constitution des équipes**, activité « Choisir mon équipe de projet » sur Moodle.
— TD PIM semaine du 20 novembre : compréhension du sujet, réponse aux questions.
— Samedi 2 décembre : **Conception du programme avec matrice pleine**. Il s'agit de définir les raffinages du programme complet en se limitant au cas d'une matrice pleine. Le traitement des arguments de la ligne de commande doit être décrit.

  On identifiera les modules qui constitueront l'application.

  Il est conseillé d'utiliser un document partagé avec votre enseignant de TD pour qu'il puisse facilement vous faire des remarques sur votre travail. Partatez-le dès le début du travail.

  Le rapport devra inclure la grille d'évaluation des raffinages complétée.
— Samedi 16 décembre : **Programme complet limité à la version version matrice pleine**, y compris la gestion des arguments de la ligne de commande, la lecture et l'écriture des fichiers.
— mardi 16 janvier : **Remise du rapport** (complet)
— mercredi 17 janvier : **Remise des autres livrables : sources, présentation et script de démonstration**
— jeudi 18 janvier : **Recette du projet**

  Elle est composée d'une *démonstration* de 5 minutes maximum, une *présentation* de 4 minutes maximum et d'un échange avec l'enseignant. La démonstration doit montrer que votre programme répond au cahier des charges. La présentation doit présenter l'architectre du programme en modules et les principaux choix techniques (se limiter aux deux choix considérés comme les plus importants). Un tirage au sort dira qui fait la démonstration et qui fait la démonstration.

5 Évaluation

Voici quelques uns des critères qui seront pris en compte lors de la notation du projet :

— la qualité du rapport

— la structure de l'application

— la qualité des raffinages

— la qualité du code (présentation, structures de contrôle adaptées, découpage en sous-programmes et modules, commentaires, etc.)

— la validité du programme

— la robustesse du programme

— les tests unitaires

Attention : Cette liste n'est pas limitative !