**Projet Info LDD2**

Modèle d’état des lieux du projet à compléter

Noms, prénoms des membres du groupe :

……………………………………………………………………………………………………………………………………………………

→ Expliciter l’approche choisie et l’état actuel pour chaque thème :

Fonctionnel / fonctionnel sur certains exemples / non fonctionnel

→ Les erreurs et points à améliorer sont à détailler dans la colonne Remarques.

→ Dans Remarque, préciser si les question bonus ont été abordées.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TD** | **Thèmes** | **Code : classe / méthode** | **Tests** | **Remarques** |
| 1 | Implémentation de graphes, manipulation | - modules/node.py : classe node (constructeur, get\_id, get\_label, parents/enfants, copy) - modules/open\_digraph.py : classe open\_digraph.\_\_init\_\_, attributs inputs, outputs, nodes |  |  |
| 2 | remove (parent, edge…), is\_wxell\_formed, add\_input\_node, add\_output\_node | - open\_digraph\_mixins/open\_digraph\_removers\_mixin.py : remove\_edge, remove\_parallel\_edges, remove\_node\_by\_id… - open\_digraph\_mixins/open\_digraph\_wellformedness\_mixin.py : assert\_is\_well\_formed, is\_well\_formed - open\_digraph\_mixins/open\_digraph\_adders\_mixin.py : add\_input\_node, add\_output\_node |  |  |
| 3 | Traductions graphe/matrice d'adjacence ; génération  aléatoire de matrice / graphe | - open\_digraph\_mixins/open\_digraph\_matrix\_mixin.py : adjacancy\_matrix, random, from\_matrix + fonctions utilitaires (random\_int\_matrix, etc.) |  |  |
| 4 | Sauvegarde (graphe vers .dot ; .dot vers graph) et représentation de graphes. Affichage (display) | - open\_digraph\_mixins/open\_digraph\_file\_display\_mixin.py : save\_as\_dot\_file, from\_dot\_file, display - modules/viz.py : vizualize\_graph (Graphviz) |  |  |
| 5 | Classe pour les circuitds booléens, méthode is\_cyclic, is\_well\_formed de  bool\_circ, shift\_indices | modules/bool\_circ.py : constructeur bool\_circ(g, debug), is\_well\_formed (avec is\_acyclic du mixin), héritage d’open\_digraph - open\_digraph.py : shift\_indices |  |  |
| 6 | Compositions & connectivité ; iparallel, icompose, identity, connected\_components d'open\_digraph | open\_digraph\_mixins/open\_digraph\_compose\_parallel\_split\_mixin.py : iparallel, parallel, icompose, compose, identity (dans open\_digraph), connected\_components, split |  |  |
| 7 | Longueur de chemins : l'algorithme de Dijkstra, shortest\_path, associer à chaque ancêtre commun des deux nœuds sa  distance à chacun des deux nœuds | open\_digraph\_mixins/open\_digraph\_dijkstra\_shortest\_mixin.py : dijkstra, shortest\_path, common\_ancestors |  |  |
| 8 | Tri topologique, chemin le plus long, mixins. | open\_digraph\_mixins/open\_digraph\_longest\_mixin.py : sort\_topologicly, get\_node\_depth, get\_graph\_depth, longest\_path |  |  |
| 9 | Synthèse de circuit via une formule propositionnelle. | modules/bool\_circ.py : fonction parse\_parentheses(...) (parser naïf de formules), build\_adder\_0 (construction directe depuis la formule) |  |  |
| 10 | Circuits booléens aléatoires ; Additionneur. | modules/bool\_circ.py : random\_bool\_circ\_from\_graph, from\_number, add\_two\_numbers, build\_adder, build\_adder\_inner, build\_half\_adder |  |  |
| 11 | Réécriture : évaluation de circuits. | modules/bool\_circ.py : méthode evaluate + toutes les transform\_\* (copy, not, and/or/xor, associativité, involution, erase, etc.) |  |  |
| 12 | Réécriture : vérification d'un code de Hamming. | bool\_circ.py :  generate\_4bit\_encoder, generate\_4bit\_decoder, carry\_lookahead\_4(n), carry\_lookahead\_4n, get\_result\_of\_evaluated\_enc\_dec | tests/encoder\_decoder\_test.py : test\_encoder\_deconder\_compose\_evaluate\_gives\_identity (et variantes), vérifie qu’après encodage+décodage et evaluate on récupère les bits d’origineb |  |