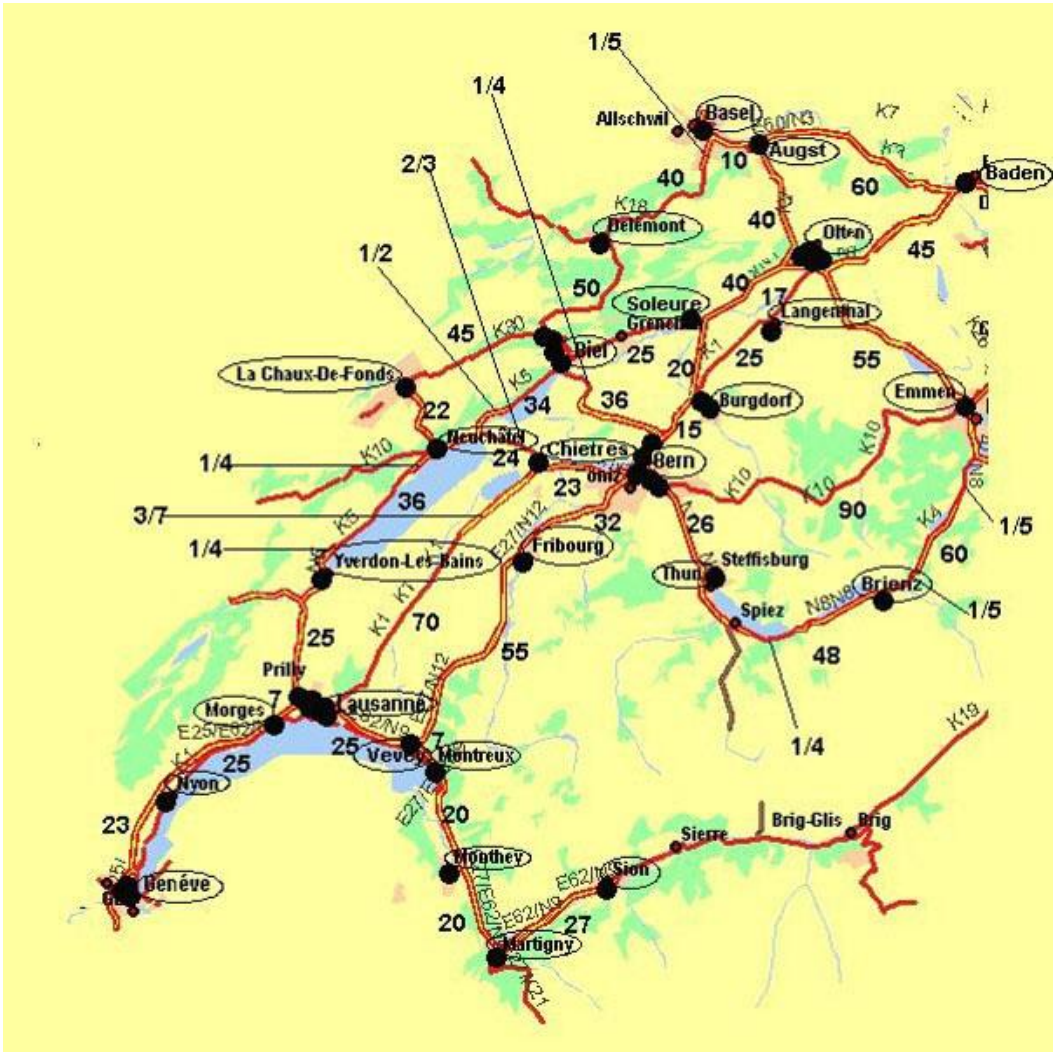
ASD 2: Labo 3

Réseau routier



ASD 2: Labo 3

Réseau routier

- Réseau routier principal de Suisse occidentale
  - Les villes sont reliées par des routes et des autoroutes (ou un mélange des deux: fraction d'autoroute)
- Application d'algorithmes de type:
  - Plus court chemin (Dijkstra à implémenter)
  - Arbre couvrant minimum
- Pour répondre à des questions du type:
  - Quel est le chemin le plus rapide entre Genève et Emmen en passant par Yverdon ?
  - Minimiser le coût de réfection des routes avec la condition que chaque ville sera accessible par une route rénovée. Prix différent selon le type de route.

ASD 2: Labo 3

Réseau routier

- Vous devrez implémenter *Dijsktra*:
  Nous vous fournissons la méthode *testShortestPath()* qui compare vos résultats avec ceux de notre implémentation de *BellmanFord.*
  Utilisation des fichiers de différentes tailles fournis

- Les temps d'exécution de *Dijsktra* (algorithme uniquement) doit rester très court
  (l'implémentation de *BellmanFord* fournie peut prendre 1-2 minutes sur le graphe à 10'000 sommets, *Dijsktra* sera plus rapide)

ASD 2: Labo 3
Wrapper

- Un *wrapper* est un objet encapsulant un autre objet dans le but de fournir une API particulière ou d'ajouter de l'abstraction (l'utilisateur passe par le *wrapper* pour accéder à l'objet «wrappé»).
- Dans ce laboratoire, il faut définir différents *wrappers* sur *RoadNetwork* (représentant le réseau routier) fournissant l'API nécessaire aux algorithmes de plus court chemin et arbre recouvrant minimum

ASD 2: Labo 3
Réseau routier

Utilisation de *Wrappers*:

```
RoadNetwork rn("reseau.txt");


RoadGraphWrapper rgw(rn);
auto mst = ASD2::MinimumSpanningTree<RoadGraphWrapper>::Kruskal(rgw);


RoadDiGraphWrapper rdgw(rn);
ASD2::DijkstraSP<RoadDiGraphWrapper> sp(rdgw, v);
```

- Ne stocker que des références !
- La fonction de coût peut être «hard-codée» mais une solution plus élégante est la bienvenue

ASD 2: Labo 3
Réseau routier

Les Wrappers doivent définir un type Edge et implémenter les méthodes suivantes:

Algorithme	V()	forEachEdge(*)	forEachAdjacentEdge(*)
Kruskal	✓	✓	
LazyPrim	✓		✓
EagerPrim	✓		✓
BellmanFordSP	✓	✓	
BellmanFordQueueSP	✓		✓
DijkstraSP	?	?	?

Egalement, les poids des arcs/arrêtes sont générés par les *Wrappers*