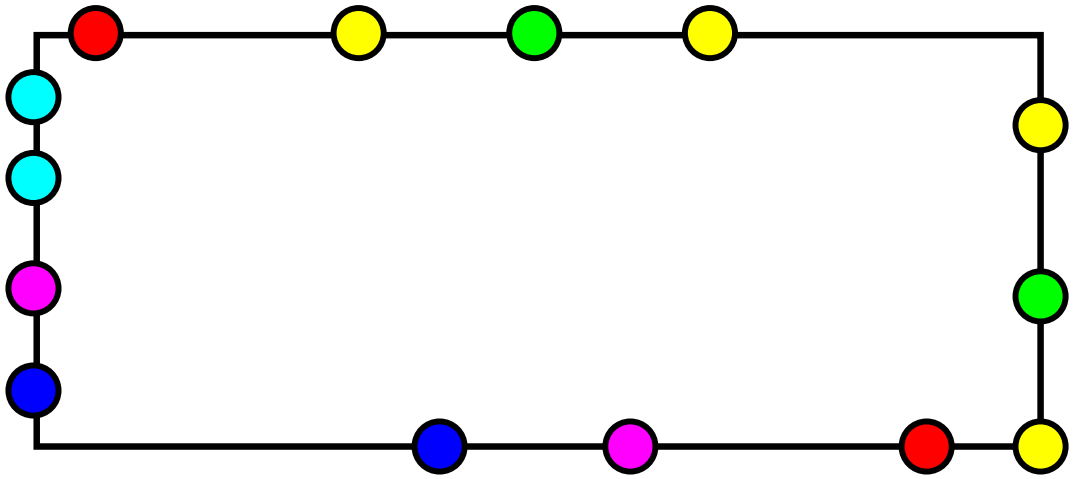
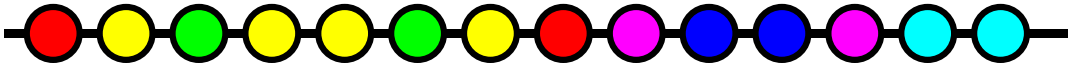
Řešení úlohy č. 1
Euklidovská kabeláž

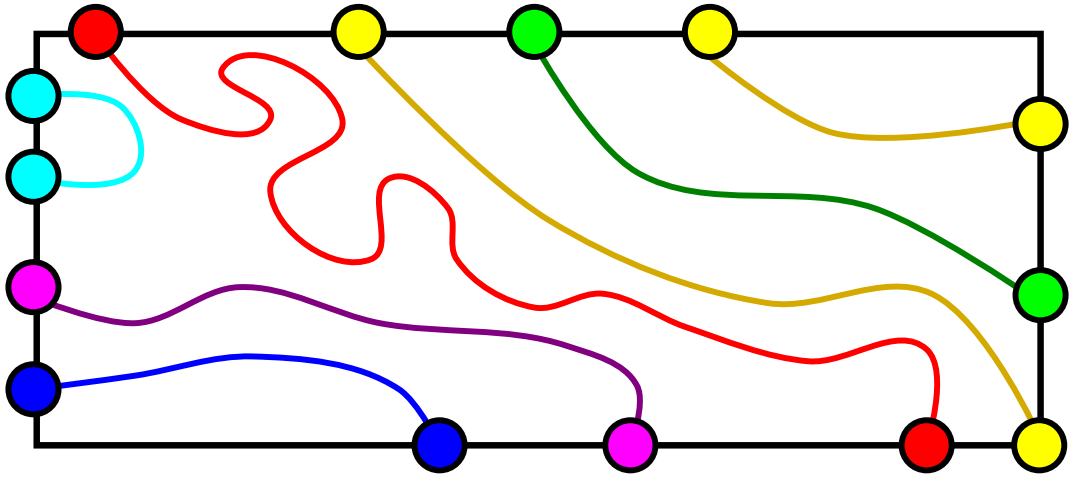
Nejprve si pojďme úlohu zjednodušit co nejvíce můžeme. Pro jednoduchost použijeme pro označení zásuvek, místo textových označení, barvy. Jeden vstup může vypadat například takto:



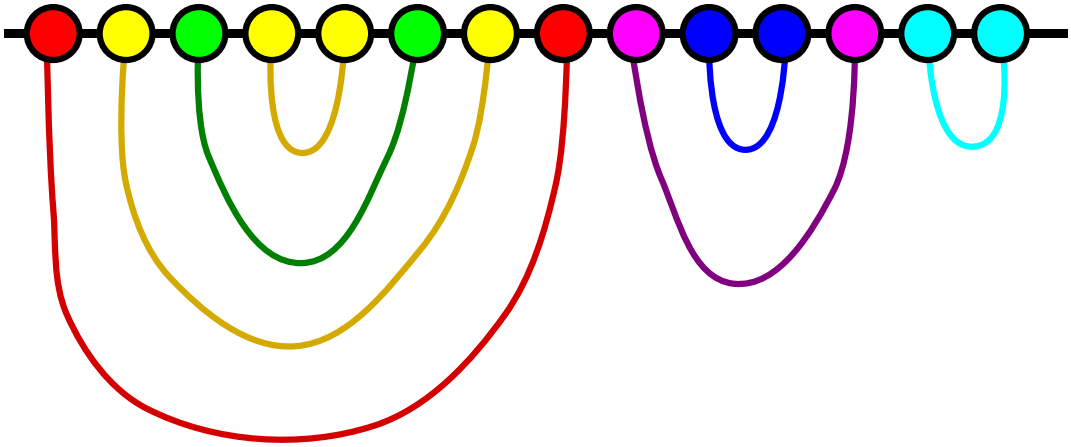
Kabely se mohou libovolně kroutit, takže první zjednodušení, kterého si můžeme všimnout je, že jediné na čem záleží, je pořadí zásuvek. Vzdálenosti mezi nimi jsou nám jedno, na které stěně leží zásuvka nás také vlastně nezajímá. Díky tomuto pozorování si lze ekvivalentně obrázek překreslit následovně:



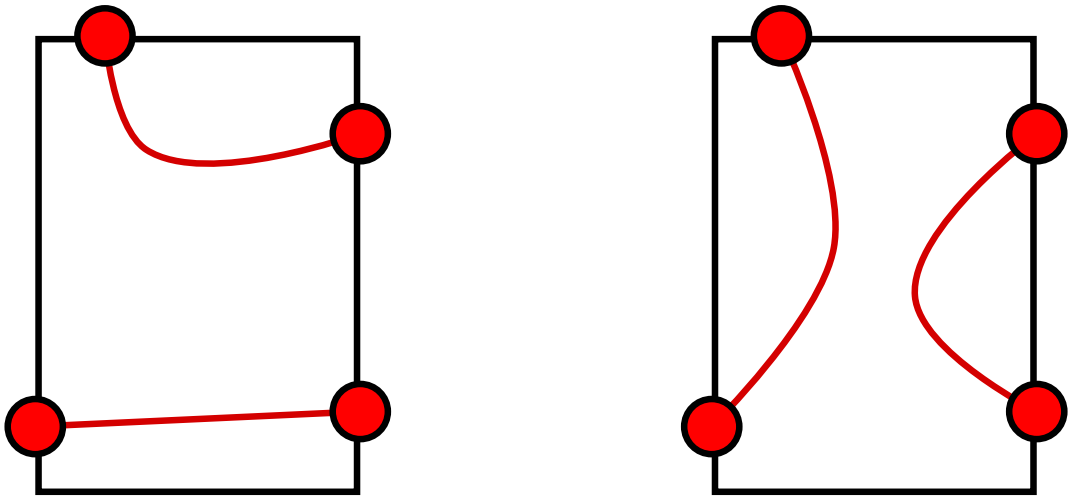
Nyní ale udělejme krok zpět a podívejme se, jak by vypadalo propojení kabelů u původního obrázku:



Jak by toto propojení kabelů vypadalo na našem zjednodušeném modelu?

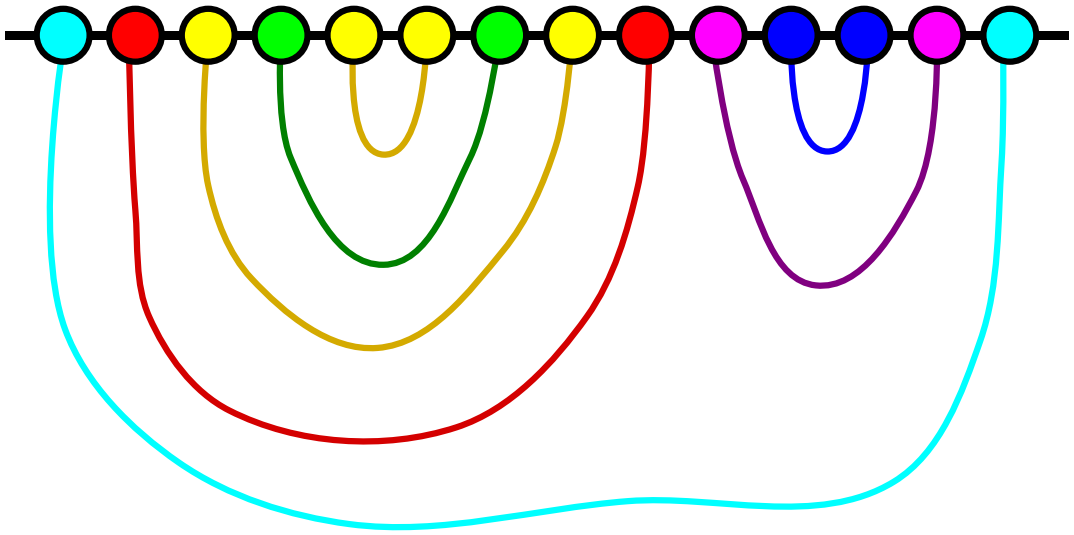


Z tohoto znázornění už lze rozhodně vypozorovat hezký vzor. Možná vás teď napadá pár otázek. Jedna z nich může být například otázka, jestli je tohle propojení jediné možné. Zrovna pro tento konkrétní příklad je tomu tak, ale pro obecný případ tomu nemusí být pravda. Zde například vidíme dvě různé validní zapojení pro stejný vstup:



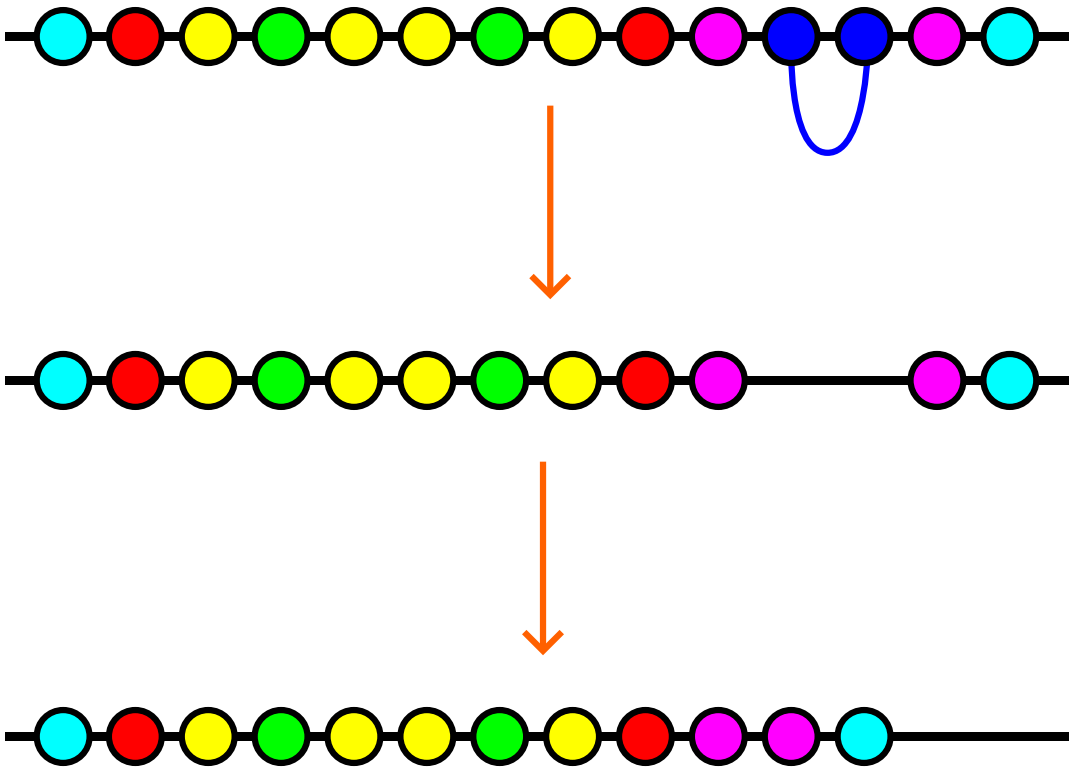
Díky tomuto příkladu víme, že propojení může být více (ekvivalentně lze říci, že jsme unikátnost řešení vyvrátili protipříkladem). Předtím, než začneme přemýšlet o algoritmu, pojďme udělat ještě následující jednoduché pozorování:

Zásuvky se chovají cyklicky. To znamená, že můžeme postupně dávat zásuvky z konce na začátek a řešení se nezmění. Už dříve jsme zjistili, že záleží jen na pořadí zásuvek za sebou, tak by nás tato vlastnost snad neměla překvapit. Níže na obrázku je například poslední zásuvka z minulého řešení přesunuta na začátek, toto zapojení je identické, pouze jsme museli vhodně přeuspořádat kabely:



Jak tedy vytvoříme algoritmus, který nalezne *nějaké* řešení, pokud existuje?

Pokud vidíme dvě zásuvky stejné barvy vedle sebe, zkazíme tím něco, když je prostě spojíme? Později si dokážeme, že opakováním tohoto pravidla opravdu nalezneme nějaké řešení, pokud existuje. Po spojení dvou zásuvek, které jsou vedle sebe si můžeme zadání zjednodušit a to tak, že tyto zásuvky prostě přestaneme uvažovat. Graficky by to vypadalo následovně:

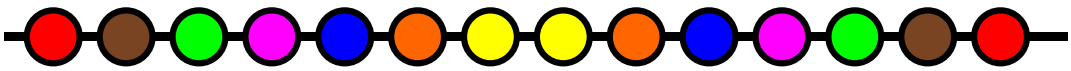


Algoritmus

```
1. Načti zásuvky do pole Z
2. Seřaď zásuvky v poli Z po obvodu, například jedním z těchto způsobů:
    * seřazením každé strany zvlášť a poté spojením 4 seřazených polí
    * měřením úhlů každé ze zásuvek do středu místnosti (tzv. pomocí centroidu)
    * klasickým seřazením s trošku delší porovnávací funkcí
3. Dokud pole Z není prázdné a po proběhnutí cyklu nějak změnilo:
    3.1 Vytvoř nové prázdné pole P
    3.2 Projdi pole Z a přidej do pole P takové zásuvky,
        které nemají hned za sebou zásuvku stejné barvy.
        Pokud má aktuálně kontrolovaná zásuvka za sebou zásuvku stejné barvy,
        přeskoč kontrolu následující zásuvky.
        (Takže obě zásuvky nepřidáme do pole P)
    3.3 Nahraď pole Z polem P
4. Pokud je pole Z prázdné vypiš "pujde to"
    (vše se napárovalo a žádné zásuvky nezbyly),
    v opačném případě vypiš "ajajaj"
```

Proč je tento algoritmus správný?
Jak moc je tento algoritmus efektivní?
Pojďme si postupně tyto dotazy zodpovědět.

Nejprve začneme s efektivitou algoritmu. Nejhorší vstup pro náš algoritmus vypadá takto (rozmyslete si proč):



Algoritmus při prvním průchodu nalezne pouze dvě žluté zásuvky vedle sebe, poté je "smaže", v dalším průchodu nalezne pouze dvě oranžové vedle sebe a tak dále…

Pokud máme zásuvek v tomto tvaru n, kolik průchodů polem náš algoritmus musí udělat, aby všechny zásuvky odstranil? V prvním průchodu projde pole délky n, v druhém pole délky $n-2$ jelikož jsme dvě odebrali, poté $n-4$ …

$$n + (n-2) + (n-4) + (n-6) + \cdots + (n-n) = \sum_{i=0}^{\frac{n}{2}} (n-2i)$$

Což se dá díky vzorci pro součet aritmetické posloupnosti zjednodušit na:

$$\sum_{i=0}^{\frac{n}{2}} (n-2i) = \frac{1}{4}(n^2+2n)$$

Všimněte si členu n^2, ten při dostatečně velkém n "přebije" všechny ostatní členy a násobně je přeroste. Velká $\mathcal{O}$ notace respektuje pouze "nejrychleji rostoucí" členy a ostatní zahazuje.

Poměrně pracně jsme tedy zjistili, že doba běhu tohoto algoritmu roste se vstupem kvadraticky (běží v čase $\mathcal{O}(n^2)$). Později si ukážeme poměrně efektivní optimalizaci tohoto algoritmu, kterému stačí pole projít pro jakýkoliv vstup pouze jednou (zlepšíme složitost na $\mathcal{O}(n)$, reálně to ale bude $\mathcal{O}(nlogn)$ kvůli řazení zásuvek po obvodu). Nejprve si ale pojďme dokázat korektnost tohoto algoritmu. Proč je spojování dvou kabelů vedle sebe vždy validní krok, který vede na řešení, pokud nějaké existuje?

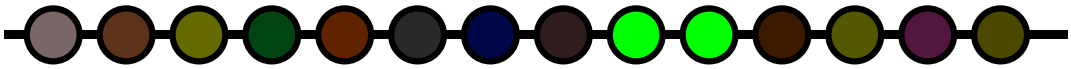
Před důkazem korektnosti si pojďme ještě více zjednodušit terminologii, s jednoduššími objekty se také jednodušeji pracuje.

Na spojení dvou sousedních stejnobarevných zásuvek kabelem můžeme také nahlížet jako na přepsání vstupu na vstup, ve kterém právě tyto dvě zásuvky chybí.

Tomuto budeme říkat *pravidlo* a původní vstup budeme značit V,
přepsaný vstup V'. Tuto akci budeme zapisovat jako $V \to V'$.

Jelikož už tu dlouho nebyl obrázek a tahle část textu začíná vypadat poměrně černobíle, tak si ukážeme jednoduchý příklad použití *pravidla*, kde se zbavíme dvou světle zelených zásuvek.

Náš původní vstup V může vypadat například takto:



Přepsaný vstup V' vypadá takto:

Provedli jsme tedy $V \to V'$.

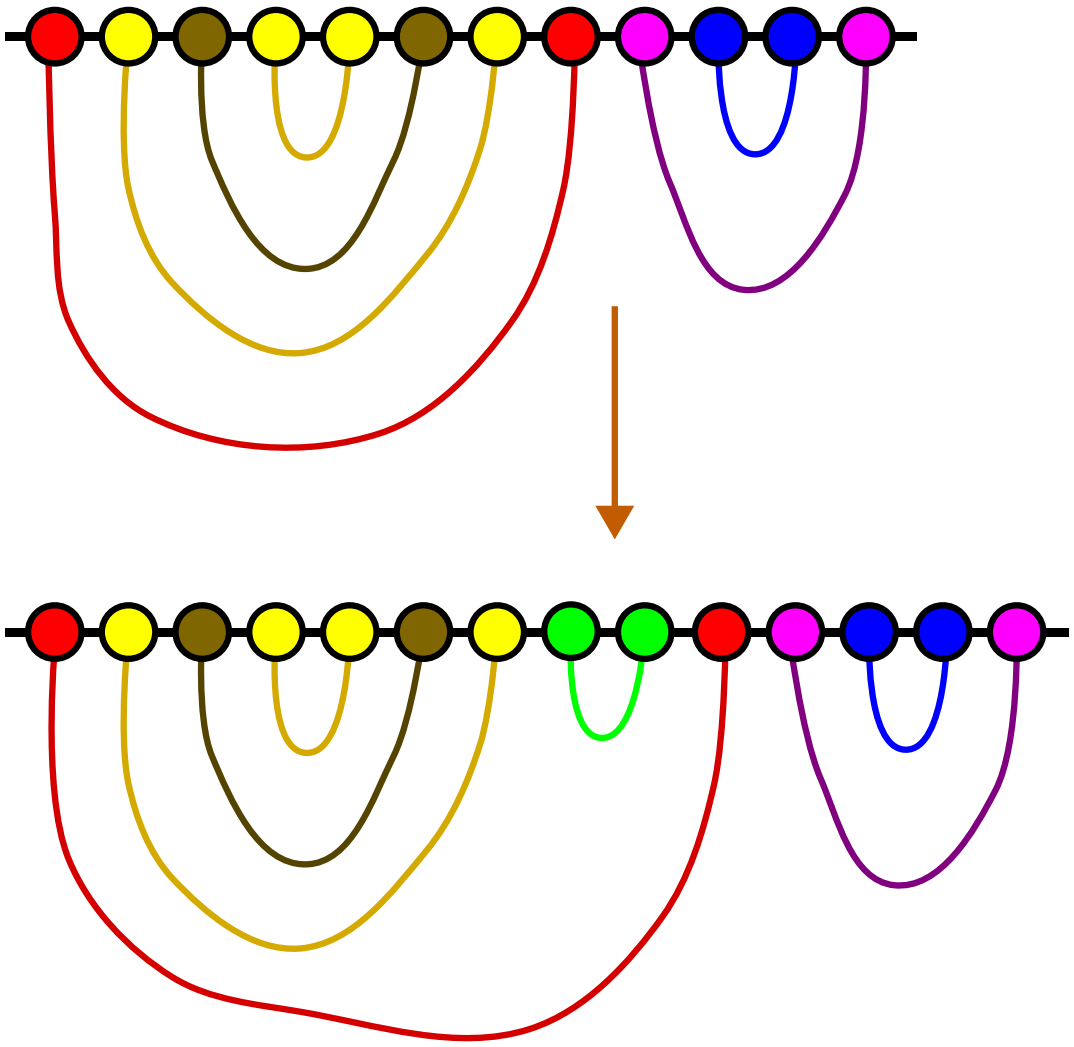
Musíme si rozmyslet, že pro důkaz korektnosti algoritmu nám stačí dokázat následující tvrzení:

Zásuvky na vstupu V jsou spárovatelné právě tehdy, když jsou spárovatelné zásuvky na vstupu V'.
symbolicky:
V **je spárovatelné** $\Leftrightarrow$ V' **je spárovatelné.**

Hospodsky řečeno, chceme dokázat, že náš algoritmus “nezkazí správnost/špatnost vstupu”.

Pro důkaz této ekvivalence budeme postupovat dokázáním dvou implikací, zprava doleva ($\Leftarrow$) a zleva doprava ($\Rightarrow$).

Pojďme tedy dokázat nejdříve jednodušší implikaci, a to zprava doleva. Chceme tedy dokázat, že
V **je spárovatelné** $\Leftarrow$ V' **je spárovatelné.**
Můžeme si rozmyslet, že pokud máme libovolné spárování, tak můžeme dvě zásuvky kamkoliv přidat a spojit je spolu, tím nepřekřížíme žádné další kabely. Graficky to vypadá následovně:

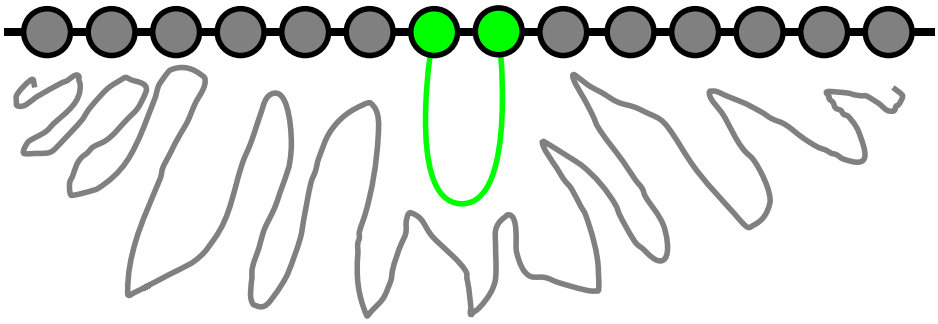


Dokázat implikaci na druhou stranu bude maličko náročnější. Pojďme si pro přehlednost opět znovu zapsat, co vlastně dokazujeme:
V **je spárovatelné** $\Rightarrow$ V' **je spárovatelné.**
Jinak řečeno, chceme dokázat, že naše spojení dvou zásuvek vedle sebe je validní (spojením nezměníme spárovatelnost/nespárovatelnost vstupu). Pojďme na to trošku sofistikovaněji a to sporem. Předpokládejme tedy, že V je spárovatelné, ale V' už není. Následně toto tvrzení rozbijeme tím, že ukážeme, že to se vlastně nemohlo stát. Tím dokážeme platnost původního tvrzení.

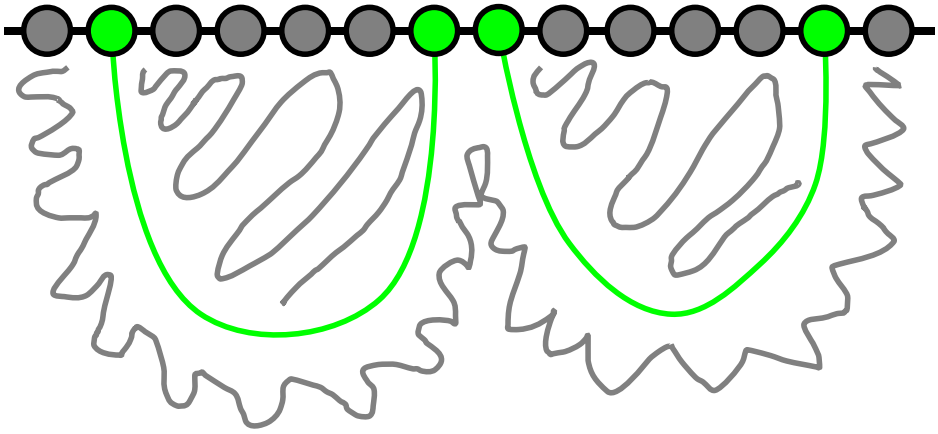
Nechť tedy po aplikování $V \to V'$ je V spárovatelné, ale V' již není. Jinak řečeno to znamená, že propojení dvou sousedních kabelů nevede na řešení, ale existuje řešení, které tyto zásuvky propojilo jinak.

Pojďme si to nakreslit. Jiné zásuvky můžeme prozatím ignorovat, všechny tedy nakreslíme šedě a šedými čárami budeme znázorňovat "nějaké propojení", detaily ostatních propojení nás v tuto chvíli příliš nezajímají.

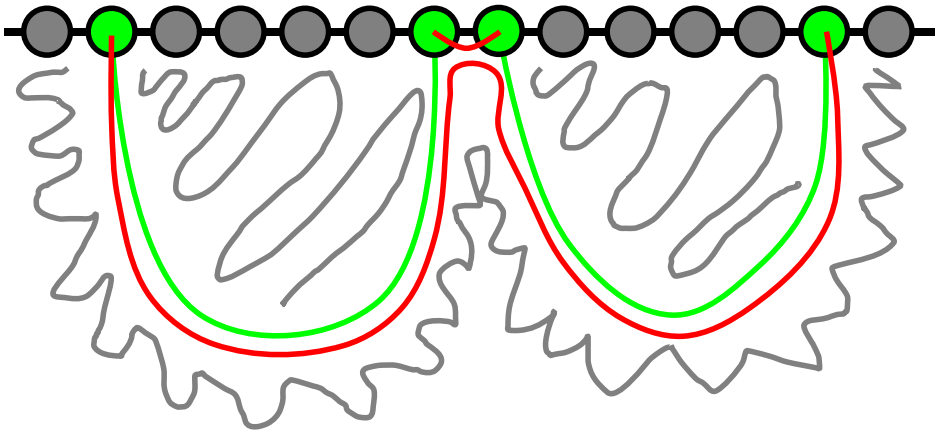
Propojení, které provedl náš algoritmus:



Propojení, které vede na správné řešení (ty samé zásuvky jsou propojeny každá s jinou další zelenou zásuvkou):



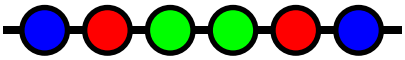
Nyní přichází jádro celého důkazu. My totiž můžeme vzít řešení, o kterém jsme předpokládali, že jsou v něm zásuvky propojitelné a to upravit tak, že spojí právě ty dvě sousední zásuvky, o kterých ale předpokládáme, že na propojitelné řešení nevedou. Tudíž s těmito předpoklady nalézáme spor, což implikuje, že se to vlastně nemohlo stát, musí tedy platit tvrzení původní, o kterém jsme předpokládali, že pravdivé není. Takové přepojení vypadá následovně (propojení co vede na řešení je značeno zeleně, přepojení je značeno červeně, i když kabel reálně červenou barvu nemá):



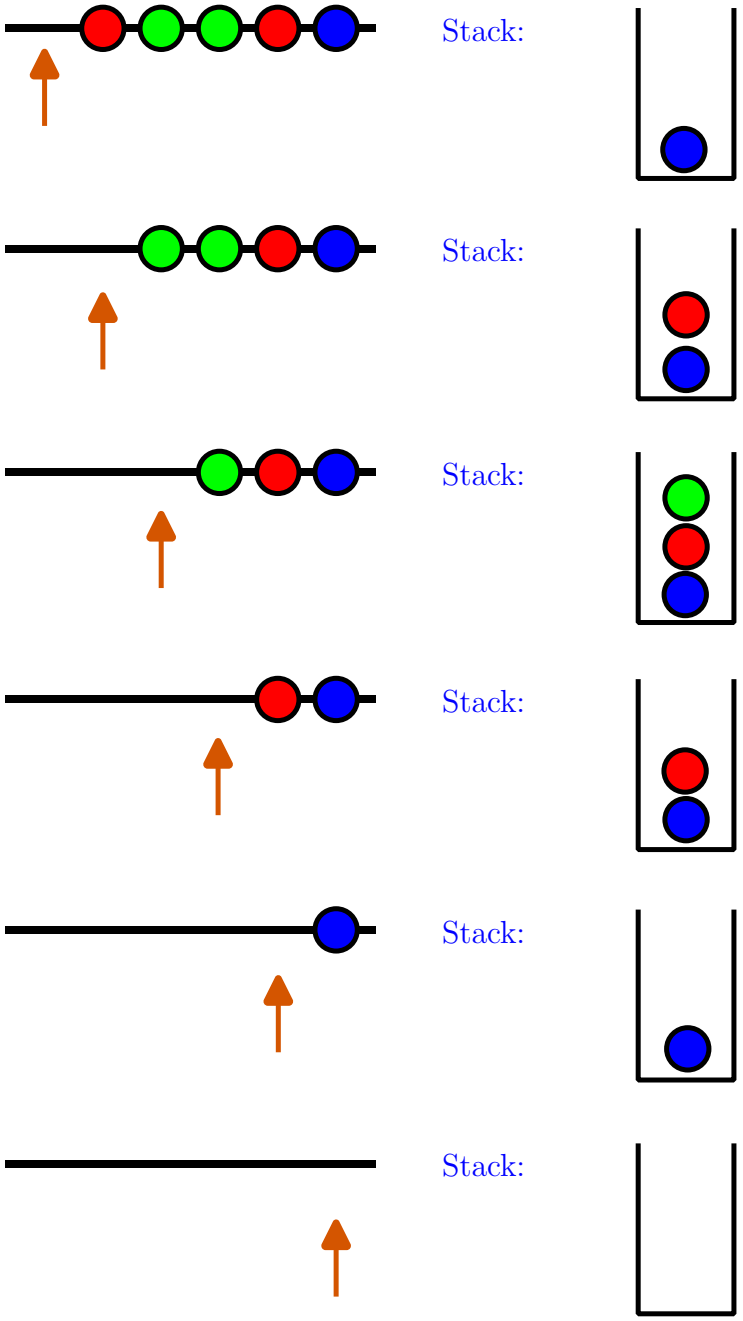
Ukázali jsme, že nalezneme takové propojení, že se žádné kabely nekříží. Náš algoritmus je tedy korektní. □

Zbývá nám udělat poslední krok. Jak zrychlíme algoritmus na slibovanou složitost $\mathcal{O}(n)$? Klíčová myšlenka je v tom, že se nechceme pořád vracet a kontrolovat zásuvky, které jsme již jednou viděli. Jedním z možných řešení je například použít datovou strukturu *zásobník*, na kterou si budeme odkládat navštívené zásuvky a pokud náhodou na vrchu zásobníku bude zásuvka stejné barvy jako se tam chystáme přidat, tak místo toho zásuvku na vrchu zásobníku smažeme (propojíme zásuvky kabelem a přestaneme je uvažovat). Tím budeme efektivně mazat zásuvky co se nachází vedle sebe a to i ty, které jsme viděli již dříve a jsou někde více hluboko ve zásobníku. Algoritmus demonstrujeme na jednoduchém příkladu:

Náš vstup V vypadá následovně:



Pojďme se podívat jak se s ním vypořádá náš vylepšený algoritmus využívající datovou strukturu zásobník:



Pokud je na konci našeho algoritmu zásobník prázdný, znamená to, že jsme zvládli vše napárovat a výstupem je `pujde to`, pokud je zásobník neprázdný, tak vypíšeme `ajajaj`.